

**User Guide**

for OpenLabyrinth version 3.0  
April 2013

This work is licensed under a Creative Commons   
Attribution-Noncommercial-Share Alike 3.0 license

# Contents

Contents 2

1. Introduction 6

1.1 What’s new in version 3.0 6

1.2 What is OpenLabyrinth? 6

2. Playing a Labyrinth 6

2.1 Open a case 6

2.2 User Interface 7

2.3 Bookmarks 8

2.4 Continuing play … 8

2.5 Finishing a case 9

3. Anatomy and Structure of OpenLabyrinth 9

3.1 More on what OpenLabyrinth is 9

3.2 Installing OpenLabyrinth 9

3.3 Where to find cases to try 10

3.4 Permissions 10

3.5 Keys 10

3.6 Navigating the Player interface 10

3.7 Labyrinths 11

3.8 Nodes 11

3.9 Links 11

3.10 Rules 12

3.11 Feedback 12

3.12 Data Elements and Clusters 12

3.13 Avatars 13

3.14 Questions 13

3.15 Chats 13

3.16 InfoButtons 13

3.17 ImageMaps and hotspots 14

3.18 Presentations 15

4. Creating a Labyrinth 15

4.1 Using the Visual Editor 15

4.2 Using Wizards 15

4.3 Creating a OpenLabyrinth Map by Importing a MedBiquitous Virtual patient Package 20

4.4 Creating an OpenLabyrinth Map using VUE 20

4.5 Creating a Labyrinth Manually 21

4.6 Creating a Labyrinth by Duplicating an existing Labyrinth 21

5. Editing in OpenLabyrinth 21

5.1 Inline Editing 21

5.2 Editor Functions 22

5.3 Edit properties 23

5.4 Nodes 24

5.5 Links 26

5.6 Play 27

5.7 Using the Visual Editor 27

5.8 Finishing a case 29

6. Advanced Authors 29

6.2 Node Grid 30

6.3 Sections 30

6.4 Feedback 31

6.5 Questions 31

6.6 Avatars 33

6.7 Files 34

6.8 Counters 36

6.9 Counter Grid 37

6.10 Counter Rules 37

6.11 Data Elements 38

6.12 Data Clusters 39

6.13 Session reports 40

6.14 Users 40

6.15 Export 40

6.16 Duplicate 40

6.17 Delete 40

6.18 Author Notes 40

6.19 Key Feature Problems and Matching 41

7. Feedback and Reporting 42

7.1 Session Reports 42

7.2 Feedback Report 43

7.3 Feedback Options 44

8. Global Functions 44

8.1 Users and Groups 44

8.2 Language support 45

8.3 Presentations 45

8.4 Collections 46

8.5 MedBiquitous Virtual Patient Export 46

8.6 MedBiquitous Virtual Patient Import 46

9. OpenLabyrinth Remote Services 47

9.1 Description 47

9.2 Setting up OpenLabyrinth Remote Services 48

9.3 Remote Services Components and Messaging 48

9.4 OpenLabyrinth Remote Services Transactions 49

9.4.1 Transaction: OpenLabyrinthservice 49

9.4.2 Transaction: OpenLabyrinth 50

9.4.3 Transaction: service 51

9.5 Basic OpenLabyrinth Client Functions 51

9.6 OpenLabyrinth Client Enhancements 52

10. Customization: Skins and Mashups 52

10.1 Skins 52

10.2 Code 53

10.3 Documentation 54

10.4 Mashups 54

11. Development Techniques 54

11.1 Using VUE 54

11.2 Using Nodes 55

11.3 Using Links 55

11.4 Using Counters 56

11.5 Different Kinds of Labyrinth Designs 56

12. Frequently Asked Questions 56

13. Further Information and Resources 57

13.1 References 57

13.2 Virtual Patient case libraries 57

14. Appendix 1: Installation 59

14.1 Preparation 59

14.2 Code and Directory 59

14.3 Database Setup 59

15. Appendix 2: GNU General Public License (GNU-GPL) v. 3.0 60

15.1 Preamble 60

15.2 TERMS AND CONDITIONS 61

16. Collaborative editing as a team 69

16.1 Potential team roles 69

16.2 General cautions about collaborative editing in OpenLabyrinth 69

17. Syntax for Counter Rules 70

17.1 Event triggers and sequence of Counter evaluations 70

17.2 Basic Syntax 70

18. ImageMaps and hotspots 71

19. Authentication systems 74

20. Connecting with other systems 74

21. Scenarios and Activity Designs 75

22. Table of Figures 75



# Introduction

## What’s new in version 3.0

The last major release of OpenLabyrinth was v2.5 released in November 2009. This new version, v3.0, includes many new and upgraded features including

The following are the main changes in version 3.0:

* Redesigned visual designer
* New conditional logic for counters and jumps
* Imagemaps with hotspots
* Revised MedBiquitous Virtual Patient import and export
* Improved questions and quizzes
* Semantic linking to mEducator2
* Authentication links to LDAP, etc

## What is OpenLabyrinth?

OpenLabyrinth is an open source online activity modelling system that allows users to build interactive ‘game-informed’ educational activities such as virtual patients, simulations, games, mazes and algorithms. It has been designed to be adaptable and simple to use while retaining a wealth of game-like features.

It has been likened to an online flexible narrative, similar to the old Choose Your Own Adventure style of book. Depending on what decisions you make or paths you navigate through the case, the consequences will be different. A well designed case should be challenging and intriguing, with real decisions similar to real-life medicine.

OpenLabyrinth is a web application that will run on most modern web browsers – you do not need any other software on your own machine. You will need access to a server running OpenLabyrinth. See section 13.2 [Virtual Patient case libraries](#_Virtual_Patient_case) for how to find such case libraries.

For more detailed notes about what an OpenLabyrinth case consists of, see section 3 on the [Anatomy and Structure of OpenLabyrinth](#_Anatomy_and_Structure).

# Playing a Labyrinth

## Open a case

This section describes the steps a user takes to start and work through an OpenLabyrinth case.

Anyone can try out any of the open cases on an OpenLabyrinth site. To start a labyrinth, just click on its title. This loads the first node and displays it (see Figure 1). At the bottom of the page, you will usually see choices on where to go next, like a Choose Your Own Adventure book. Simply click on the link to go to that page.

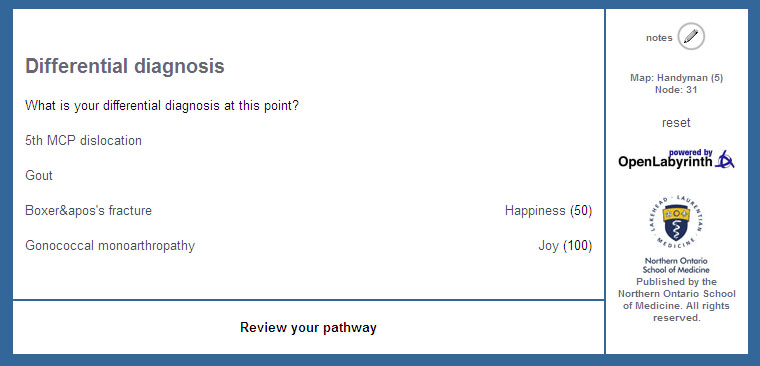


Figure 1*:* a typical labyrinth screen

To get more out of most OpenLabyrinth sites, you need a login id. This will give you access to more cases, and also to better feedback reports at the end of your case. You can also try out your hand at writing your own simple cases. [Contact us](mailto:info@openlabyrinth.ca?subject=Request%20for%20trial%20login%20(via%20OLab%20manual)) if you would like a trial [login on our demo site](http://demo.openlabyrinth.ca).

## User Interface

There may be a number of different elements on display (see Figure 2: a typical labyrinth screen’s elements) including:

* Title: every node has a title, which is typically displayed at the top of the page.
* Node content: although not mandatory almost all labyrinth nodes will have some text content for the user. Typically that will describe the consequences of having made the previous choices of paths and possibly things to consider in making the next choice.
* Linked options: the way to navigate an OpenLabyrinth case is to click on one of the available options here. These will usually show the title of the node you are inking to.
* Review your pathway: a clickable track of your pathway through the labyrinth since you started is available. Click on a link to go back to that point – note that this does not roll back the track of which nodes have been visited. Every action is recorded in a session. This may also affect how some Counters and Rules operate.
* Counters: a simple way to show your progress in a game. Authors can show or hide counters on each node, minimizing distracting information when not needed. There may also be a display of how the counter’s value has changed since the last node.
* Map and Node ID: this shows the current labyrinth and node ID number. If there is a problem with a case, the case authors will appreciate you making a note of which node ID number caused your problems.
* Reset link: this ends your present session and restarts you in a new session within the same OpenLabyrinth case.
* A link to OpenLabyrinth home page: clicking the OpenLabyrinth icon returns you to the home page.
* Other graphics, links, tools and text may also be displayed depending on the current skin.

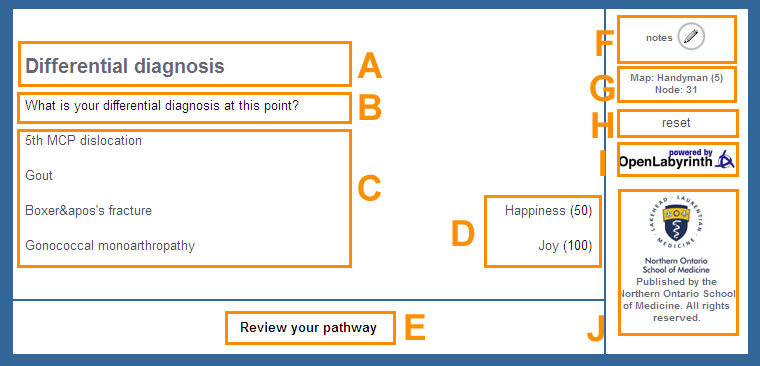


Figure 2: a typical labyrinth screen’s elements

A: Node title; B: Node message/content; C: Links/options; D: Counters – label plus current value; E: Link to review nodes viewed in the current session; F: Open note taker; G: Current labyrinth and node information; H: Reset/restart current labyrinth; I: link to OpenLabyrinth home page; J: Skin-specific graphics and text.

## Bookmarks

Sometimes a labyrinth may be too long to complete in one session or perhaps a learner only has a little time. The bookmark feature (enabled in the Basic skin only by default) allows you to set a bookmark at a certain point within a labyrinth. To restart from a stored bookmark go to “Labyrinths I have played” from the home page, then select the labyrinth and then click the “bookmark at ‘node name’” link to carry on from that point. Note that only the last bookmark in any given session is stored. At present, this is not yet functional within OpenLabyrinth v3.

## Continuing play …

Generally, navigating your way through a well-designed OpenLabyrinth case should be reasonably intuitive. Many cases follow a fairly linear format, often in the HEIDR (History, Exam, Investigation, Diagnosis, Rx or management) model. But OpenLabyrinth authors are not at all restricted to this style. OpenLabyrinth provides an environment in which cases can have many different styles – multiple endings, varying levels of complexity, and can support some simple games where the flow is determined by the game designer. See Section 11.5 for some more information about case designs and topology.

What you experience depends on the design of the labyrinth they are playing and the choices available within it but typically you make decisions as to which path they shall take and these decisions have different consequences depending on the nodes they pass through. Scores and counters may go up or down, paths may be dead ends or choices may end the current activity, while other paths will be successful. When you are playing a labyrinth you should think carefully about the options available to you, keep an eye on any counters or timers in the activity and follow the instructions and hints given you.

## Finishing a case

Most cases will make it clear when you have finished (saved your patient, killed them etc). You might be invited to reflect on your scores in the Counters. Some cases will provide the option to have a more detailed report on your performance. You must be logged in to the OpenLabyrinth server at the start of the case for this.

Every option selection (or ‘click’ - along with the current score, timer and counter values) is recorded as you work through a labyrinth. This tracking supports the pathway review function as well as being able to generate a report on how you did within the activity.

If the current labyrinth has been set to provide a feedback report then at some point you will be presented a link that says “**end session and view report**” – this will end the current session and provide a comprehensive feedback report – see section 7 [Feedback and Reporting](#_Feedback_and_Reporting) for more information.

# Anatomy and Structure of OpenLabyrinth

## More on what OpenLabyrinth is

As a player of cases, you don’t really need to understand what is going on behind the scenes. Often, it is better to hide the complexity of a case behind a nice simple interface. You simply need to work your way through a case, clicking on links, answering questions and quizzes. But if you plan to write your own cases, it does help to understand the basic anatomy of an OpenLabyrinth virtual patient case.

OpenLabyrinth is an open source online activity modelling system that allows users to build interactive ‘game-informed’ educational activities such as virtual patients, simulations, games, mazes and algorithms. It has been designed to be adaptable and simple to use while retaining a wealth of game-like features. OpenLabyrinth is licensed under the GNU General Public License v3 (GNU-GPL3 : [www.gnu.org/licenses/](http://www.gnu.org/licenses/)) - see Appendix 2 in section 15

The original Labyrinth application was originally developed by the Learning Technology Section of the College of Medicine and Veterinary Medicine at the University of Edinburgh, which was subsequently revised into OpenLabyrinth by Rachel Ellaway at the Northern Ontario School of Medicine along with input from partners in Canada, the UK, Germany, Australia and the US.

OpenLabyrinth v3 was completely rewritten under the guidance of the OpenLabyrinth3 Consortium, lead by the University of Calgary, as an open-source project. The code was completely rewritten in PHP, obviating the previous problems arising from dependencies on closed 3rd party code libraries. The main version now is based on Linux, Apache, MySQL and PHP, rather than using Microsoft Windows servers and software. Because the source code is openly available, it can be recompiled for similar platforms, including Windows servers. Further information about this is available in section 14 [Appendix 1: Installation](#_Appendix_1:_Installation).

This guide is intended to act both as user guide and technical documentation for the system.

## Installing OpenLabyrinth

OpenLabyrinth is a web application that will run on most modern web browsers – you do not need any other software on your own machine. You will need access to a server running Apache and MySQL, the commonest platform for web-based applications. For demonstration and test purposes, you can request access to our demo server at <http://demo.openlabyrinth.ca> by emailing us at [info@openlabyrinth.ca](mailto:info@openlabyrinth.ca?subject=Request%20for%20author%20access%20on%20demo.olab3%20(from%20user%20guide)) - this server runs the most recent version of the OpenLabyrinth code. It is not recommended by production level cases. For this we recommend that your organization sets up its own server. See [Appendix 1: Installation](#_Appendix_1:_Installation_1) for more details on how to install and configure OpenLabyrinth.

## Where to find cases to try

There are many libraries of OpenLabyrinth virtual patient cases available where you can try running a case to see if it might fit your needs. Firstly, try some of the open cases available at our main server, <http://vp.openlabyrinth.ca/> or see section 13.2 [Virtual Patient case libraries](#_Virtual_Patient_case_1). A more updated list is available at our web site: <http://openlabyrinth.ca> - please let us know if you come across other accessible cases that are not on our list. Many of these libraries will feature older version 2 OpenLabyrinth cases but these will still give you an idea of what can be done with this application. OpenLabyrinth v3 retains full compatibility with the ANSI/[Medbiq Virtual Patient (MVP) standard](http://www.medbiq.org/std_specs/standards/index.html#MVP), which means that it will run OpenLabyrinth v2 cases, and also virtual patients from other virtual patient engines that are compliant with the MVP standard.

Some cases that we recommend you look at first include:

* [VP on VPs](http://demo.openlabyrinth.ca/renderLabyrinth/index/49) – a virtual patient on virtual patients
* [Welcome to OpenLabyrinth](http://demo.openlabyrinth.ca/renderLabyrinth/index/5)
* [Harriet Headcase](http://vp.openlabyrinth.ca/renderLabyrinth/index/53) – a case designed in 3 parts for group work
* [IUCD Mashup](http://vp.openlabyrinth.ca/renderLabyrinth/index/10) – illustrating how OpenLabyrinth can be used to splice video segments
* [Chester Angermeier](http://vp.openlabyrinth.ca/renderLabyrinth/index/21) – an example of a case ported from the eViP repository
* [Mildred Blonde](http://vp.openlabyrinth.ca/renderLabyrinth/index/23) – example of a vague and dizzy historian
* [John's back again](http://demo.openlabyrinth.ca/renderLabyrinth/index/33) – written by one of our residents
* [Teaching Tips](http://demo.openlabyrinth.ca/renderLabyrinth/index/35) – a placeholder for linked information. Not a patient.
* [Counting on You!](http://demo.openlabyrinth.ca/renderLabyrinth/index/44) – showing how counters can be used. For authors.
* [Medical Careers](http://vp.openlabyrinth.ca/renderLabyrinth/index/45) – based on a classic board game

## Permissions

The only labyrinths you can run without logging in are those whose security has been set to ‘open’. Labyrinths with a security setting of ‘closed’ can be viewed by anyone logged into OpenLabyrinth while ‘private’ maps can only be viewed by their logged in editors.

## Keys

In addition to making a Labyrinth more or less secure using Labyrinth security you can also require your users to enter an arbitrary text key to activate a Labyrinth. To turn on the use of keys set the labyrinth security type to “keys (a key is required to access this Labyrinth)” in the global editor and use the ‘edit’ link by the security type select to create one or more keys. A key can be any kind of text string including variations such as ‘green for go’ and ‘85A94W8BA9445’.

When a user tries to run a labyrinth they will be challenged to enter a valid key and they won’t be able to start until they do so.

## Navigating the Player interface

This has already been described – see section 2 [Playing a Labyrinth](#_Playing_a_Labyrinth).

## Labyrinths

A ‘labyrinth’ is the principal unit of organisation within OpenLabyrinth. Each labyrinth has a series of global properties such as the type (game, maze, algorithm etc), its authors, timers, visual appearance (skins), security, scores and counters etc. Within each labyrinth there are a series of linked pages or ‘nodes’ that define the options available to you, each of which can be enhanced with a number of behaviours and services to further structure your experience and gameplay.

## Nodes

A labyrinth’s nodes are the unit of presentation to the user, each one displaying as a web page, around which all the other labyrinth components are organised. A case will consist of many interconnected nodes. Figure 3 shows a diagrammatic representation of a simple activity consisting of six nodes.

Every labyrinth node has a unique identifier (node ID) as well as a number of other properties such as a title, textual content, a type (root or child) and a series of rule and function properties. The rules and functions determine what is presented to you at any given node depending both on current node properties and what you have done previously (particularly concerning scores and counters). Other node properties include whether it is the root node (the labyrinth starting point), whether the node must be visited or avoided and whether you can end the session and see a report of how they did.

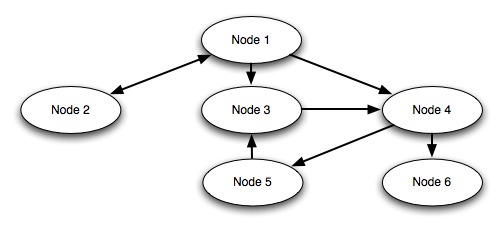


Figure 3: nodes and links

*Node 1 is linked to nodes 2, 3 and 4, node 2 is linked back to node 1, node 3 is linked to node 4, node 4 is linked to nodes 5 and 6, node 5 is linked to node 3 and node 6 isn’t linked to anything.*

## Links

Labyrinth nodes are connected by a series of links expressed as pairs of node IDs with some additional properties such as ordering, icons and alternative text. Because links are one directional (from node A to node B) a link back requires a second link (from node B to node A) with independent properties. There are 8 links indicated in figure 3 because there is a link from node 1 to 2 and another separate one from node 2 to 1.

Links can be presented to you in four different ways

* Hypertext – clickable text link per available choice
* Drop down – a dropdown list of the available choices
* Dropdown with confidence interval – as above with a second drop down indicating how confident (or unconfident) you are about your choice
* Type in text – you type in the first few letters of your proposed answer and if it matches an available option this is auto filled for you.

The order of these available options can be set, randomized or randomly set to present just the one option.

## Rules

Rules are functions attached to nodes that change the way a labyrinth is presented to you. One set of rules might set or change one or more counters while another could require you to have visited other nodes before being able to load the current one. In a little more detail these rules include:

* Counters: there can be as many or as few counters in a Labyrinth as you want. Each counter has a name, description, starting value and a number of functions. At the entry to any node or on clicking any link the value for each and every counter can be changed using plus, minus or equals operators along with an integer value; ‘+10’ adds ten, ‘=4’ sets the value to 4 irrespective of its previous value. These values can then trigger rules based on the current value of each counter.
* Conditionals: these control access to a node based on which nodes you has visited previously. For instance a rule that looks like “{15}OR{16}” would mean that you couldn’t enter node 17 without having visited nodes 15 or 16 first. Each rule is made up of node IDs connected with standard Boolean operators.
* Timers: these provide a real time countdown at the end of which you needs to restart the activity. A 300 second timer would mean that you would need to complete the activity in 5 minutes. Timer functions are not yet fully implemented in OpenLabyrinth v3.

In OpenLabyrinth v3, we have instituted a more powerful approach that allows the case author to apply some simple IF.. THEN.. ELSE logical structures to how the case is portrayed. This is potentially very powerful and has already opened up some interesting development possibilities for us. The syntax is inherently simple but see section 17 [Syntax for Counter Rules](#_Syntax_for_Counter) for a more detailed description and how to use them.

## Feedback

The user can be given extensive feedback based on their choices in playing a labyrinth. This includes a report of which choices were made, the counter values and whether nodes were marked as ‘must visit’ or ‘must avoid’.

There are a number of author-configured feedback rules including:

* Feedback per node visited
* Feedback depending on the numbers of ‘must visit’ and ‘must avoid’ nodes visited
* Feedback on the time taken to complete
* Feedback on values of counters at the end of the session

## Data Elements and Clusters

OpenLabyrinth has been designed to import to and export from the MedBiquitous virtual patient data standard, an emerging specification for the exchange and reuse of virtual patient activities between different authoring and player systems. For more information, see section 6.11 [Data Elements](#_Data_Elements)

## Avatars

One way of enhancing a narrative is to identify human characters or agents within its flow. OpenLabyrinth supports the use of characters by providing support for simple animated avatars for these characters in the narrative. An avatar can be configured to appear differently (age, skin, hair, clothing, context) as well as communicate using speech or thought bubbles. The same avatar can be reused within different settings or many different characters can be used.

## Questions

OpenLabyrinth users can control how an activity plays out by selecting from the available choices along the way. Sometimes however some interaction is needed that doesn’t advance the story. Questions can be embedded within a node. These can either be open text entry or multiple choice, the latter producing a score and controlling a counter.

## Chats

These are similar to Questions in appearance. They are simple way of providing extra information to you, if they click on them. They can also control a Counter.

## InfoButtons

These are also a simple way of providing additional information on a node or page. They appear to you as a simple blue “i” logo. 

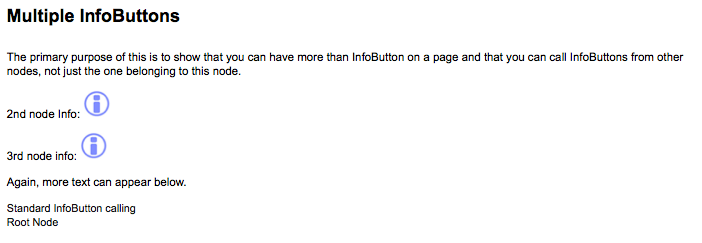


Figure 4: a node can show more than one InfoButton

When you click on the icon, a small sub-window appears with additional information. InfoButtons are generally connected to their parent Node but you can call an InfoButton from another Node.

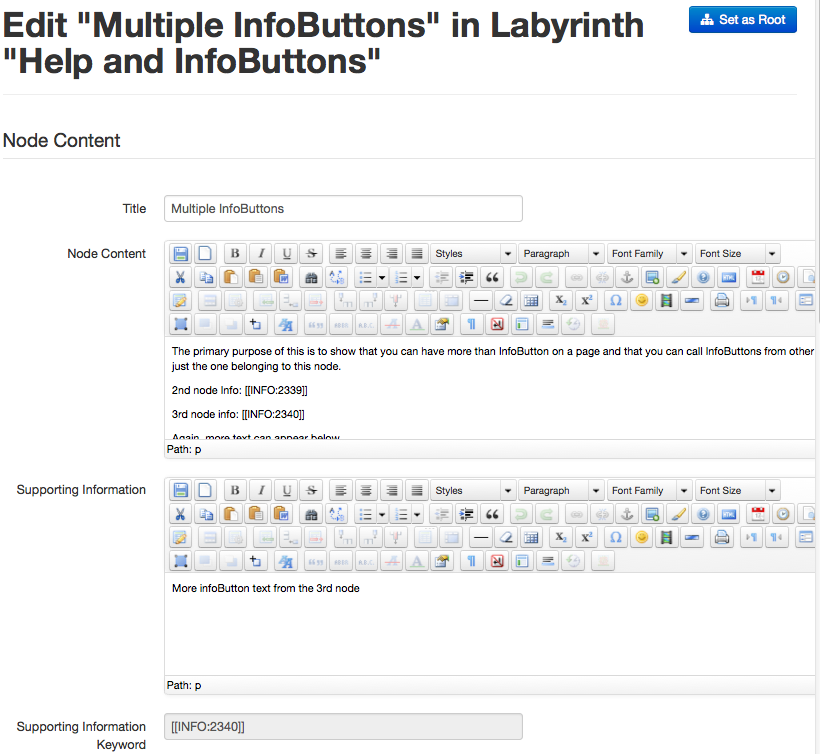


Figure 5: edit InfoButtons using the Supporting Information field

Note that you can place the infoButton anywhere within the Node Content. Use the [[INFO:xxx]] style of reference, just as you would for images and Questions. Unlike OpenLabyrinth v2, where you only had to place some text in the Supporting Information field, you still have to place this marker or else the infoButton will not show up.

## ImageMaps and hotspots

Within the Node Editor, OpenLabyrinth now gives you the ability to create hotspots on images that act the same as links. These can be links to other nodes, just as with any other OpenLabyrinth link, or can be hyperlinks to any web address. This can be used to extend the visual metaphor of the OpenLabyrinth interface, providing things like virtual EMR screens, clickable hotspots on x-rays or other diagnostic images, or any other clinical image where you want to the learner to select a choice. See section 18 [ImageMaps and hotspots](#_ImageMaps_and_hotspots) for more details on how to set up an imagemap.

## Presentations

A presentation is an arbitrary set of labyrinths grouped together for users to access together. This may be a set of materials for a particular course, an examination paper or a group topic.

# Creating a Labyrinth

There are several ways to create a labyrinth within OpenLabyrinth. You must be logged in as an author on your OpenLabyrinth web site. If you do not have a local OpenLabyrinth server running, [contact us regarding a guest login](mailto:info@openlabyrinth.ca?subject=Guest%20login%20for%20OLab3%20authoring) on one of our systems.

## Using the Visual Editor

This is a simple concept-mapping tool, similar to the VUE concept-mapping tool from Tufts, which we previously used for case design. The OpenLabyrinth Visual Editor is now completely integrated into the rest of the OpenLabyrinth authoring environment. See this section for more on how to do this – [Using the Visual Editor](#_Using_the_Visual)

## Using Wizards

There is now a set of simple wizards that can help you with the initial steps in designing a case. More experienced authors might find it quicker to use more direct methods but this does help to get new authors with getting set up with basic structures of a case.

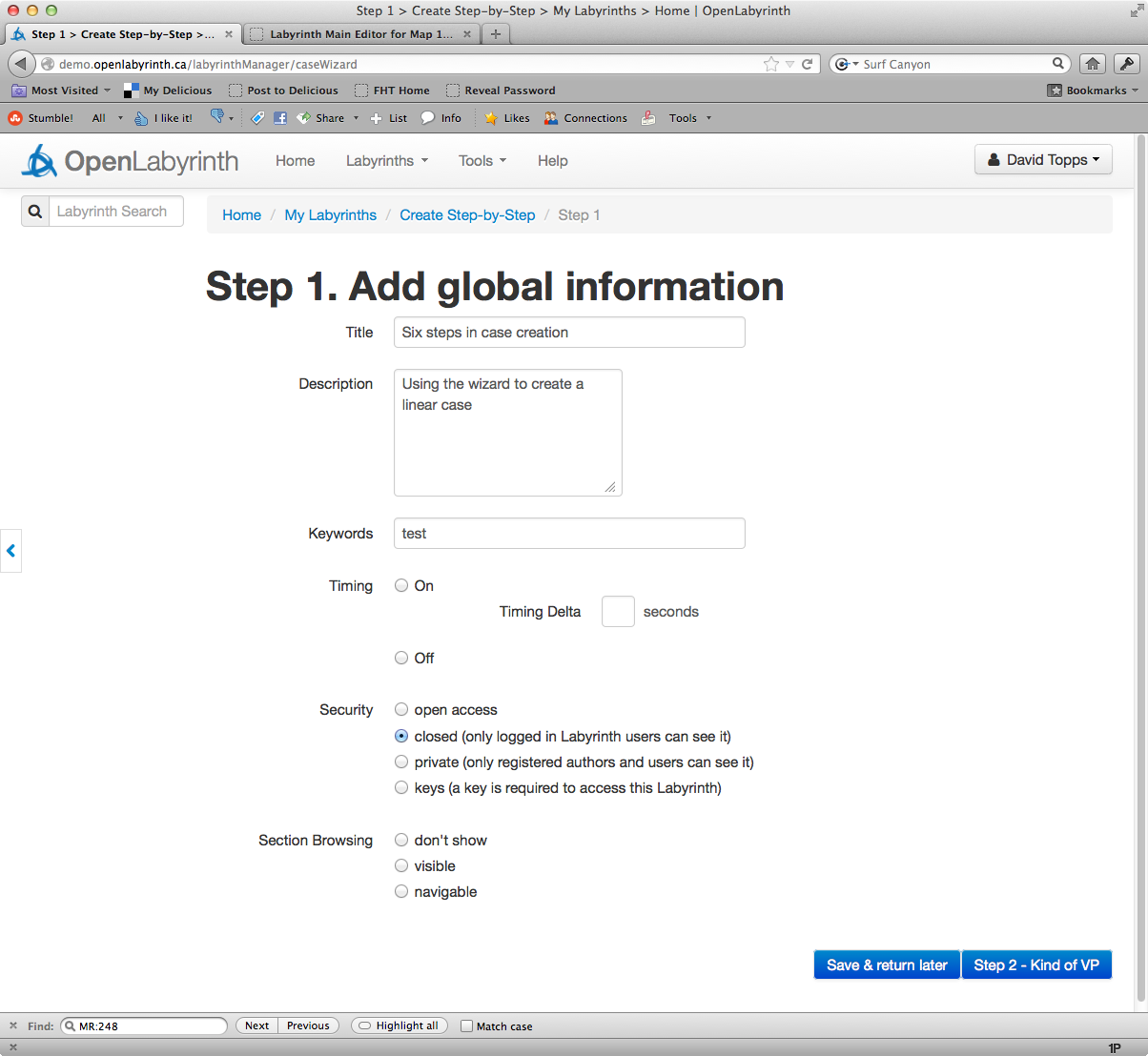


Figure 6: first step in wizard to Create a case

To get started with case design using a wizard, select ‘Create Step-by-Step’ from the Labyrinths top menu. You will first be asked for global information for the case. All of the information you enter here and subsequently can be edited later so don’t worry too much about getting things right just now. Add a title, short description of your case, and appropriate keywords if you wish. You can ignore Timing choices for now. Click on ‘private’ for Security. Move on to Step 2.

In Step 2, choose what pattern or learning design you want to use. The commonest to start with is a linear or HEIDR case but the power of OpenLabyrinth lies in branched cases. See section 11.5 for examples of some common pattern designs. For this example, we will choose a Linear pattern.

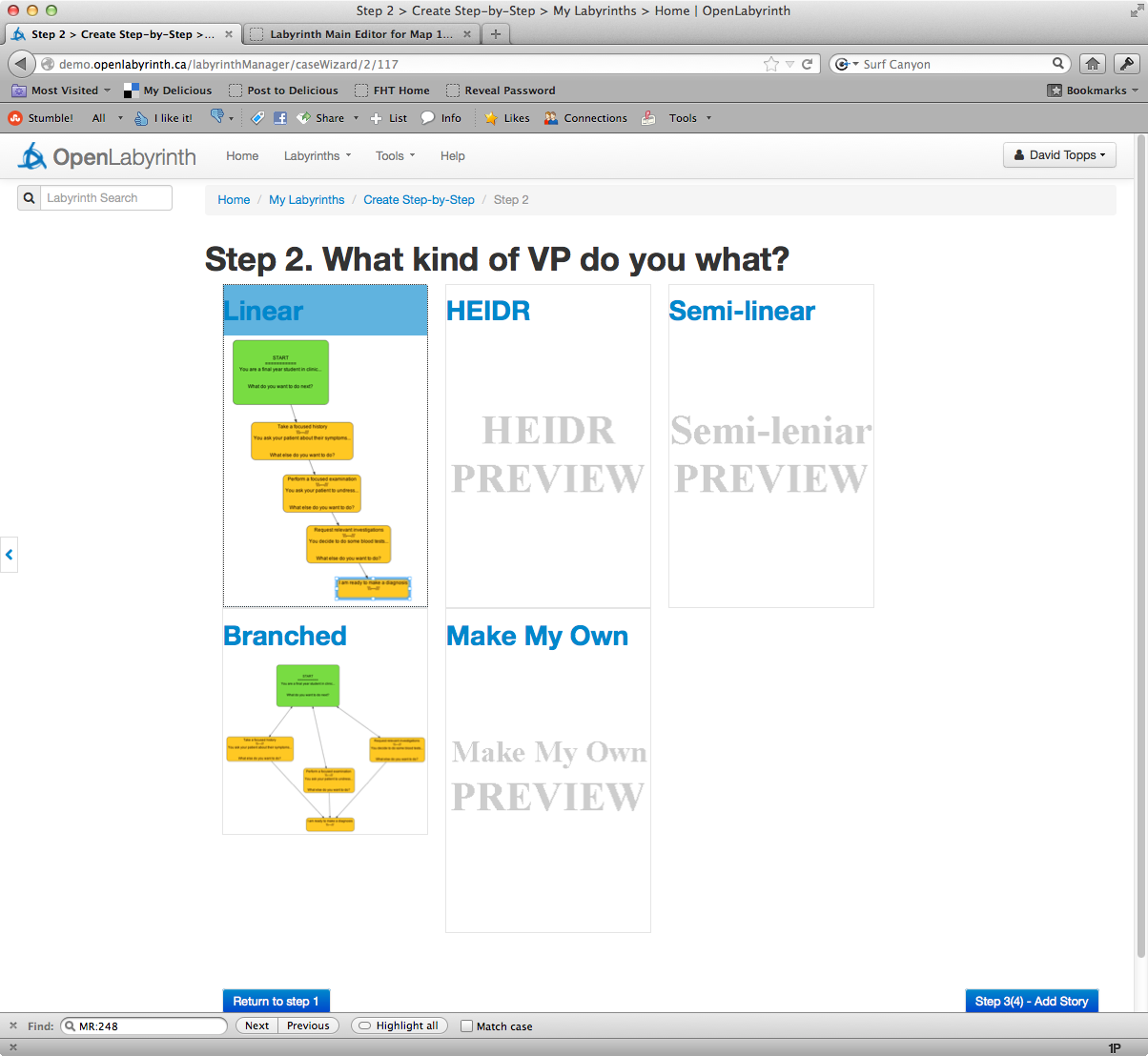


Figure 7: select the kind of pattern you want to use

Go to Step 3 to add your story parts. In this step, we will flesh out the key points to get across. We have chosen a HEIDR style of case, typical for an undergraduate medical presentation: History, Exam, Investigations, Diagnosis and Treatment (Rx). It is good at this point to have a clear idea of the 3 to 4 main teaching points that you want to illustrate. Enter these in the nodes you see here. You can always come back later to flesh out the story more, provide a greater variety of choices, more context and background etc, but getting these main points down now in the backbone of your story will help to keep things focused.

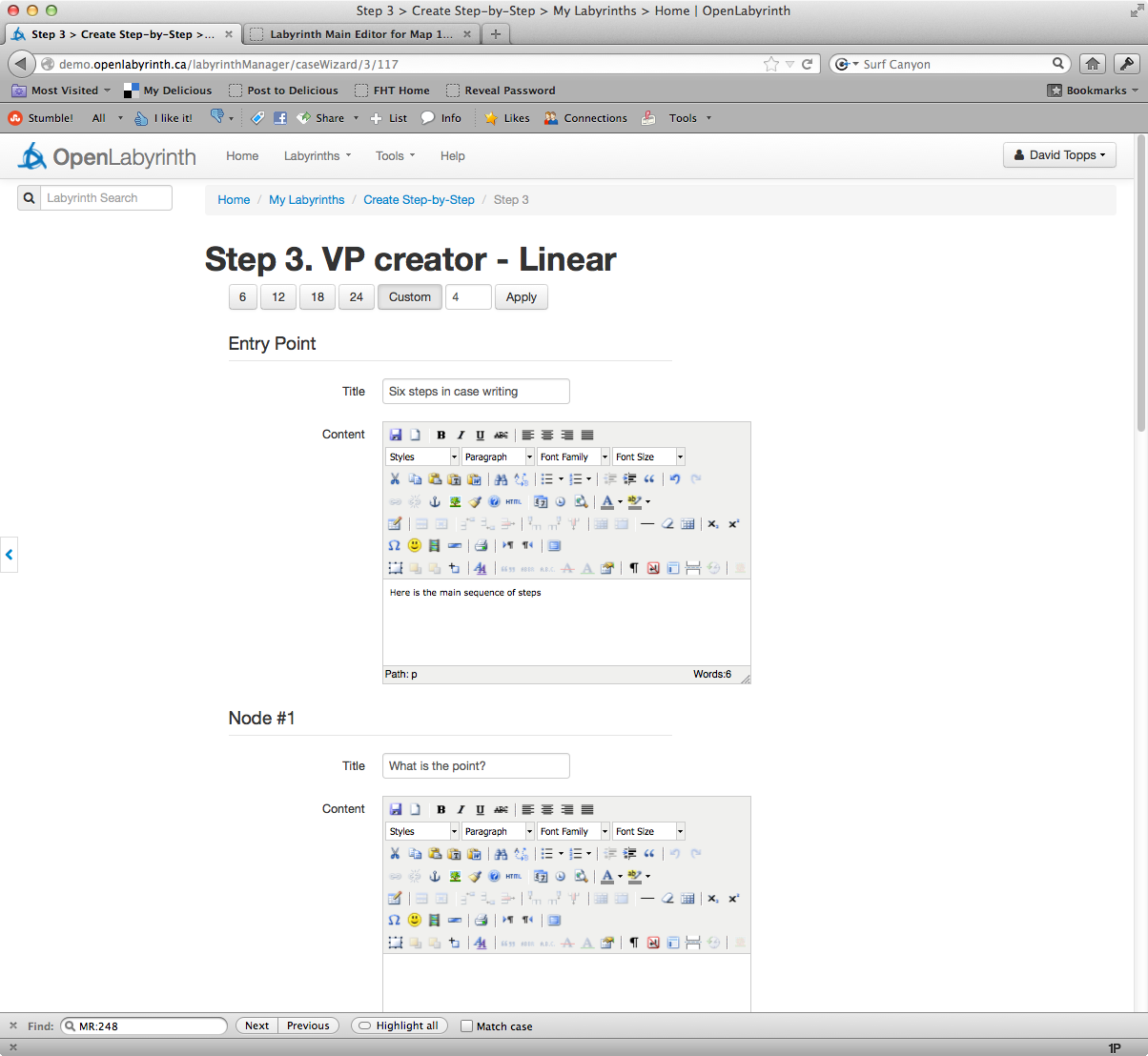


Figure 8: Select number of nodes then fill in bare essentials

Fill out the nodes in Step 3 as an overview then proceed to Step 4 to edit your story points further.

Click on the blue button at bottom right to proceed to Step 4. This will bring up the Visual Editor, showing you your linear pattern, with the information you just inserted.

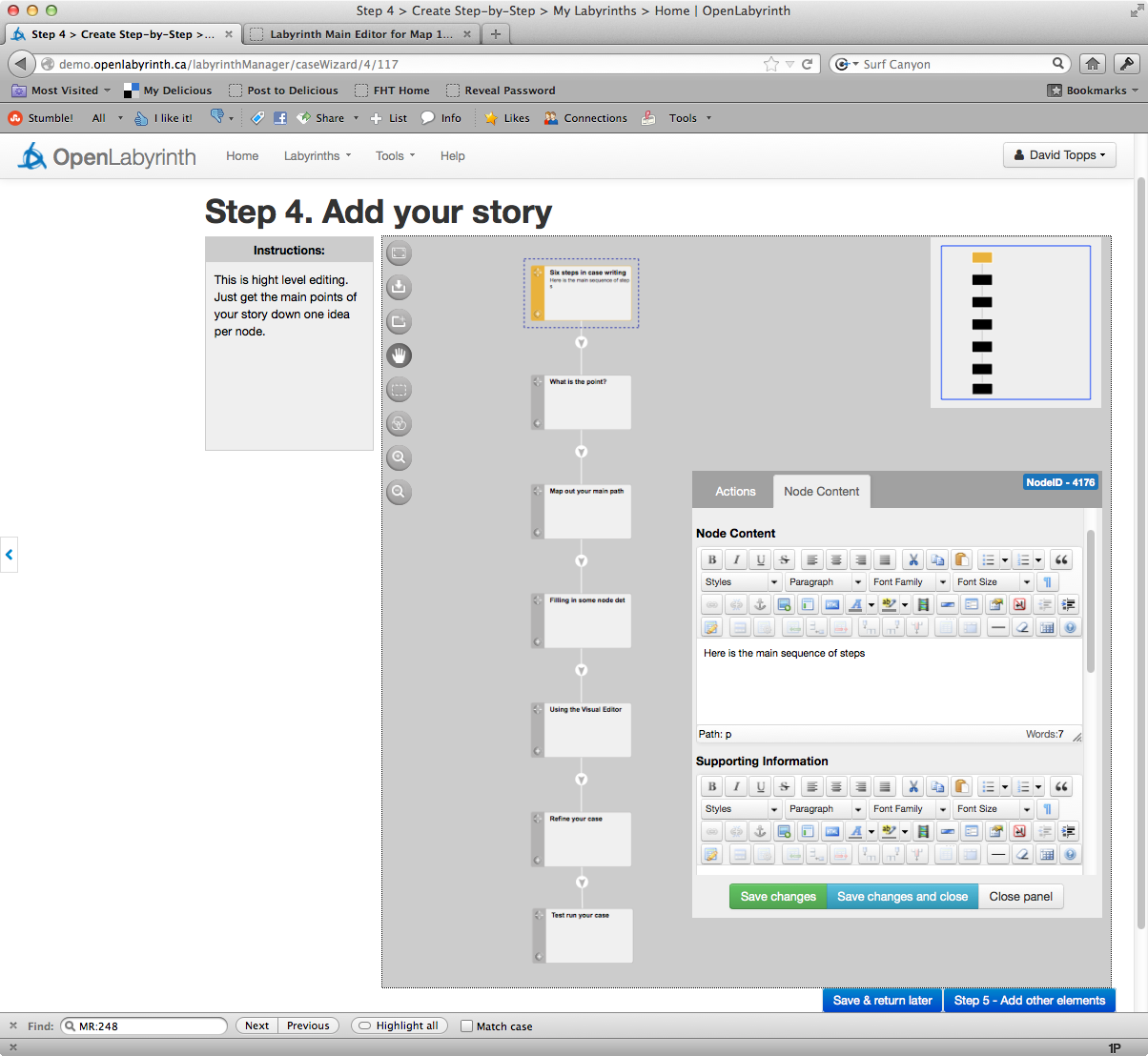


Figure 9: using the Visual Editor to continue mapping the pattern

Add a few additional side nodes for your case here. Don’t feel you have to do it all now. You can come back and add more later.

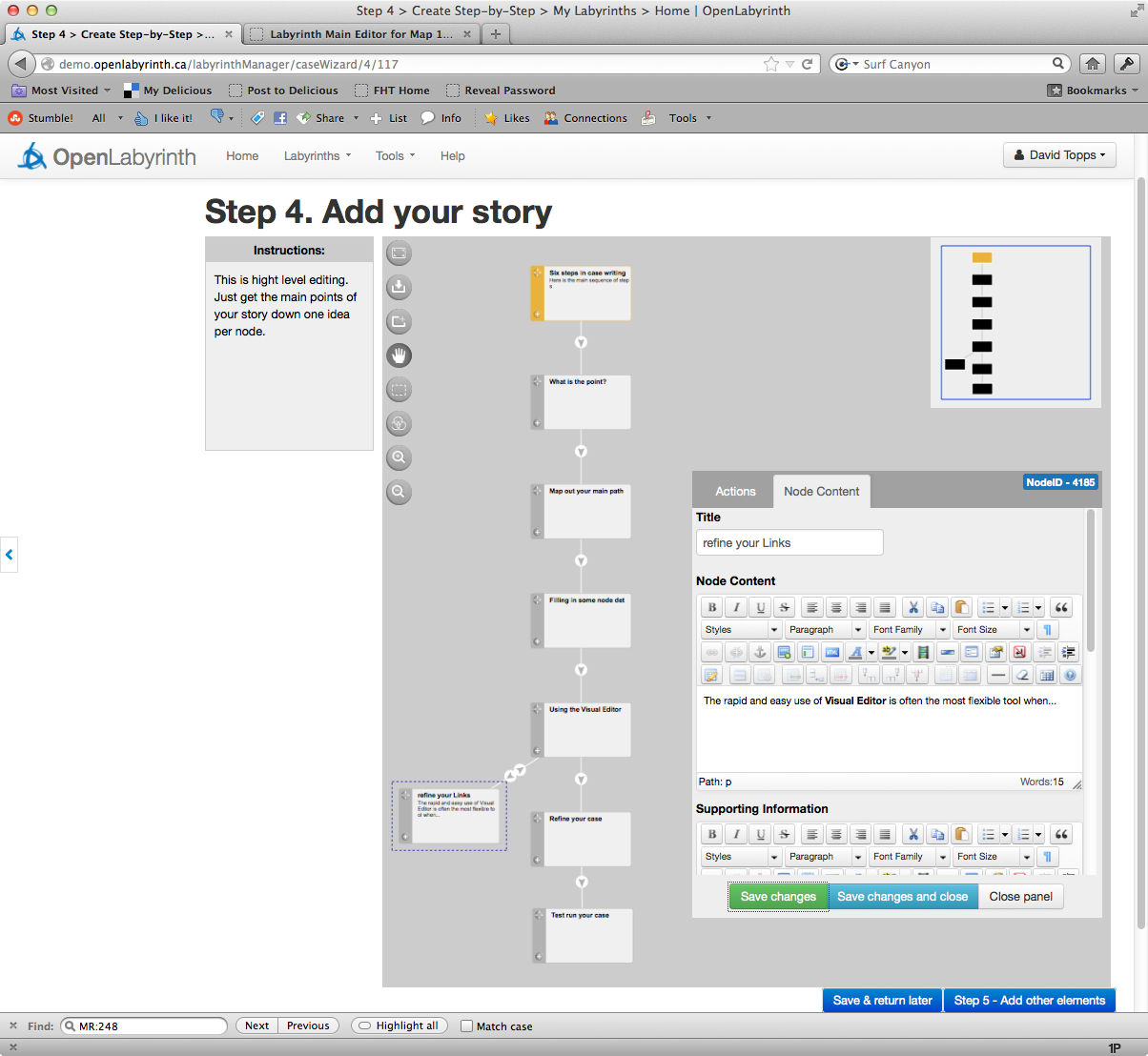


Figure 10: adding a small side branch using Visual Editor

In Step 5, you can add other parts to your case such as pictures, avatars, or insert some brief multiple-choice type questions.

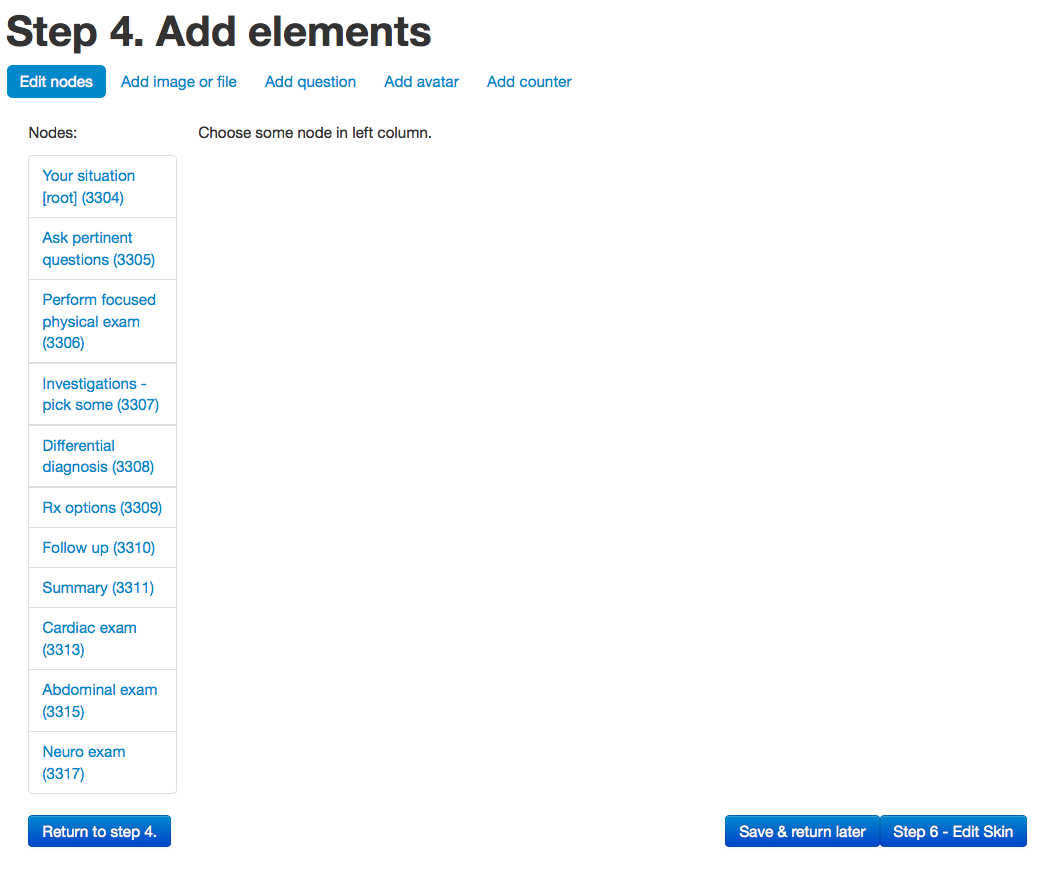


Figure 11: add remaining components here

In the final step, you can create or select a Skin that fits the theme of your case. You can also do this at any time in the future. For now, you might want to stick with the basic skin and focus on your case content.

Click on ‘Save & return later’. You can now go back to edit your case further using a variety of editing tools. See section 5 [Editing in OpenLabyrinth](#_Editing_in_OpenLabyrinth) for more details on how to do this.

## Creating a OpenLabyrinth Map by Importing a MedBiquitous Virtual patient Package

OpenLabyrinth can create new labyrinths by importing MedBiquitous Virtual Patient packages. The package and the import process are described more fully in [MedBiquitous Virtual Patient Import](#_MedBiquitous_Virtual_Patient).

## Creating an OpenLabyrinth Map using VUE

VUE is a Java-based visual concept-mapping tool from Tufts University that can be downloaded (for free) from http://vue.tufts.edu for both Windows and Mac. You can use it to create designs for labyrinths by creating boxes to represent nodes and the links between them. Although VUE supports many other features, only the boxes (converted to nodes), text in the boxes/nodes and the links (between nodes) will be imported. Everything else will be ignored when the VUE map is imported into OpenLabyrinth.

This feature is **not** currently directly supported in OpenLabyrinth v3. You can still use VUE but the process is now more convoluted. You import your VUE map into OpenLabyrinth v2, export as an MVP format file and then import the MVP file into OpenLabyrinth. Please consult the [OpenLabyrinth v2 manual](http://pine.nosm.ca/documents/userguide.pdf) on how to best use VUE to construct a map that can be smoothly imported.

## Creating a Labyrinth Manually

This is where each node is entered manually, one at a time, along with the links, rules etc. The steps for authoring and editing are very similar other than for authoring you need to create a labyrinth first. To do this first click the ‘add a new Labyrinth’ link (available to logged in users only) and then fill in the form to set the new labyrinth’s global properties.

Once you have created the new labyrinth you can then start building it up by adding nodes, links between nodes and counters, files, rules and other dynamic properties. A single root node will have been created as part of the labyrinth but all other nodes, links, rules etc will need to be added manually – see section on [Editing in OpenLabyrinth](#_Editing_in_OpenLabyrinth_1) for more details on how to do this

## Creating a Labyrinth by Duplicating an existing Labyrinth

Any labyrinth you have edit access to can be duplicated as a template for a new labyrinth by clicking its ‘duplicate’ link on the editor page. A basic copy of the original labyrinth is created (called “Copy of …”) allowing you to change any aspect while keeping the original untouched. Note that duplicating just takes the basic structure and does not import files, rules or other additional properties.

# Editing in OpenLabyrinth

All aspects of a labyrinth can be edited by anyone who is logged in with edit permissions for that particular labyrinth. Simple content editing can be carried out within a node as the labyrinth is being played using the inline editor, while more advanced features are available through the other labyrinth editors.

## Inline Editing

The content of a node can be edited from within that node while a case is being played, if the current user has authoring rights to the current labyrinth. Click the link saying ‘turn editing on’ to switch into inline editor mode. You will be able to change the current node’s title and content from here or click directly to the more advanced editing tools (see figure 12).

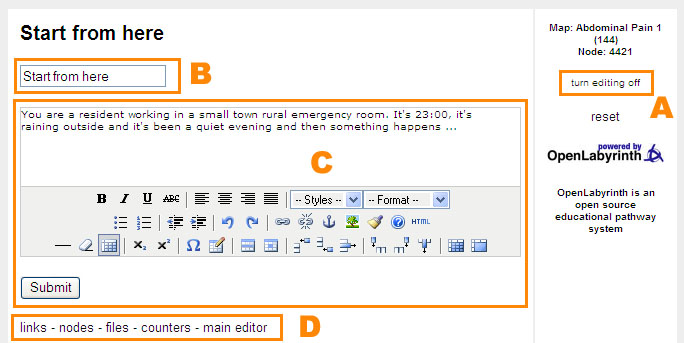


Figure 12: inline node editing:

*A is the link to turn editing on and off, B is the field to change the title, C is the field to change the text, and D is a series of links to the main editor tools*

## Editor Functions

When you open your case, the first page that opens is the ‘Edit properties’ page

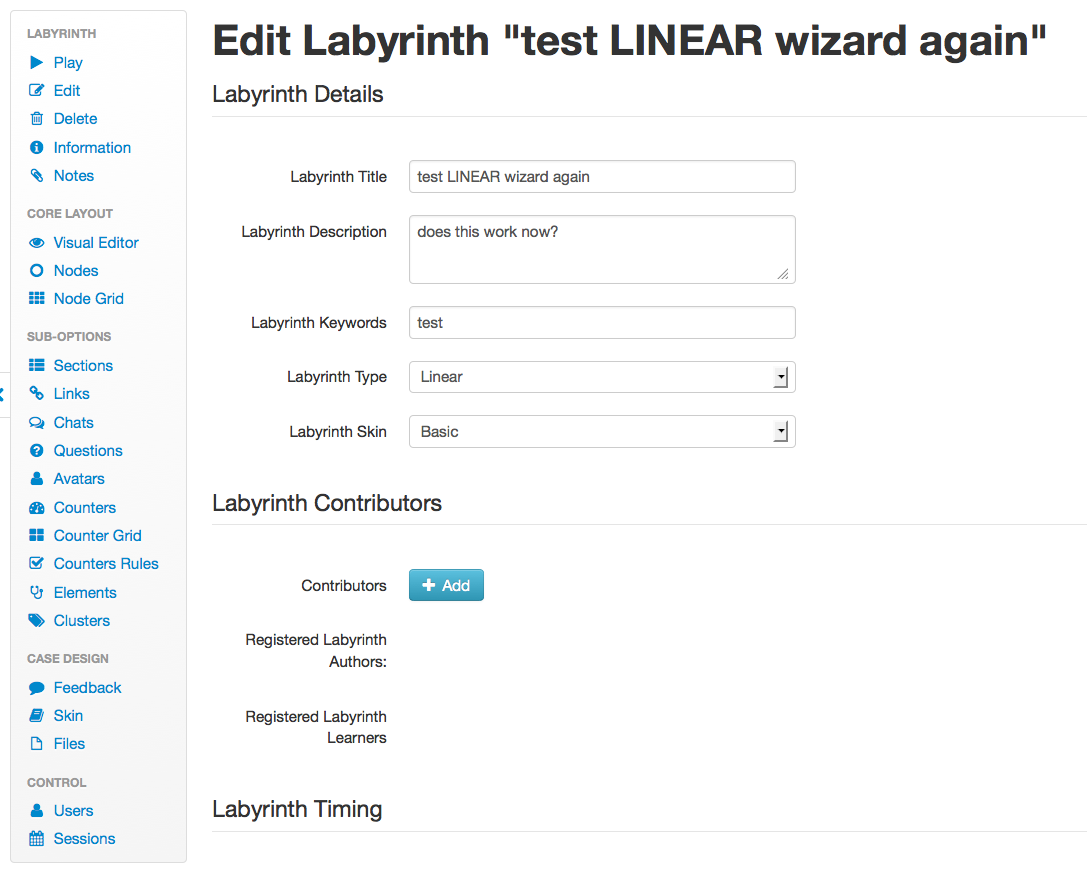


Figure 13: the OpenLabyrinth properties editor, with common actions in sidebar on Left side.

There are many main editing functions available in the left panel:

* Play – this launches the map to run in a new window
* Edit properties - this is a duplicate of the form used to create a map manually
* Delete – deletes the current labyrinth
* Information – some statistics about the case.
* Notes – a small notepad for the case authors. This is quite limited at present.
* Visual Editor – visual node and link editor, in concept-mapping style
* Nodes editor – you can add nodes or edit any existing nodes from here
* Node Grid – you can edit all of the node titles and contents within a single Labyrinth at once
* Sections – you can create arbitrary sections for grouping nodes together and assign nodes to these sections from here.
* Links – you can add links or edit links from here
* Chats -
* Questions – embed questions for tracking and evaluation
* Avatars - add and edit different avatars for the current labyrinth
* Counters – you can add and edit counters from here
* Counter Grid – you can see and edit all counter functions for all nodes in a Labyrinth at once
* Counter Rules – you can use simple IF..THEN..ELSE type logic to alter how the case plays.
* Elements – you can add and edit data elements from here
* Clusters – you can add and edit data clusters from here
* Feedback – you can create and edit feedback rules from here
* Skins – you can alter how the case appears, with differing colors, fonts, backgrounds and layouts.
* Files – you can upload and manage any files you need in your map from here
* Users – you can add or remove users (both authors and learners) for the current Labyrinth from here
* Sessions – reports on all user sessions for the current Labyrinth - see section [Session reports](#_Session_reports)

## Edit properties

These are the properties that structure the whole labyrinth including:

* Title, author credits, keywords and description: these are free text entries to set general metadata properties for this particular map.
* Labyrinth type:
  + Maze and Algorithm – these only use one message box so that the use of scores is not advised for these labyrinth types, counters should be used instead. Maze is the default type.
  + Game – uses scores to control messages –deprecated from v2.08 onward
  + Key feature problem – enables additional assessment functionality
* Skin selection: a labyrinth can be skinned to appear in a range of different visual styles – you can select from any of the existing skins. Note that some skins have some OpenLabyrinth features disabled. See section 10 on customizing OpenLabyrinth for more information on developing and using skins.
* Timing: whether the map uses a real timer and the time delta (time available) set in seconds (integers only). \*\*This feature is coming soon.
* Security: can be set – see table 1: security settings and their meaning

|  |  |  |  |
| --- | --- | --- | --- |
| Security type | View/run | Edit | Duplicate |
| Open | Yes | Only if logged in and registered as an editor for the particular labyrinth | Only if logged in and registered as an editor for the particular labyrinth |
| Closed | Only if logged in |
| Private | Only if logged in as an editor for the particular labyrinth |
| Key | Requires a text-based key to run |

Table 1: security settings and their meaning

* Section browsing: this controls how the node sections are shown to you. The choices are: don't show, visible, navigable (clickable).

## Nodes

To edit the nodes in the current map, click the ‘Nodes’ link in the main editor.

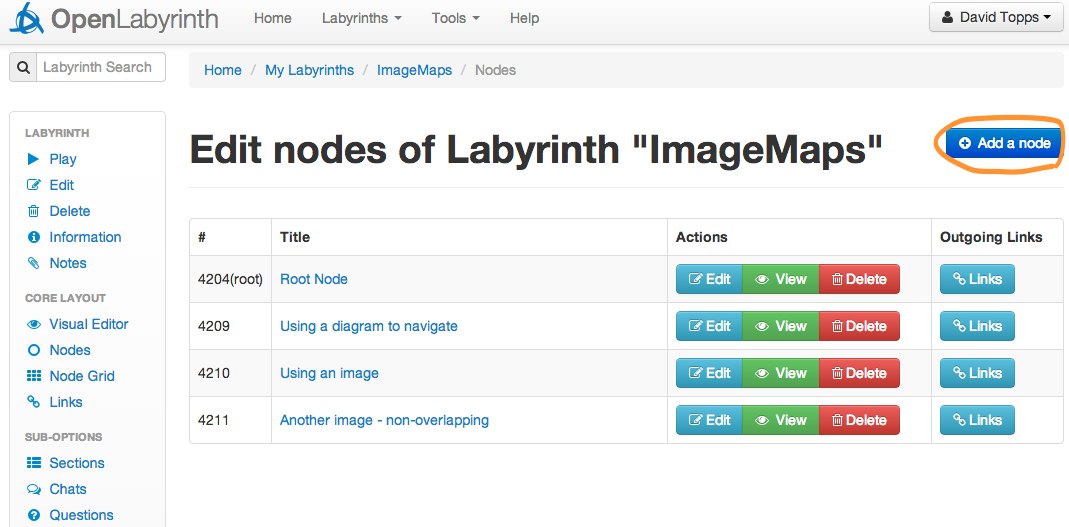


Figure 14: Nodes panel with 'Add a node' button highlighted

This node edit screen lists all of the nodes in the current map. Clicking on the title of a node will preview it. Clicking on ‘edit’ will launch the node editor and clicking on ‘Links’ will launch the link editor for that node. There is a button at the top of the node listing page to let you add a new node.

Remember to click on the ‘Save changes’ button at the bottom of the page, once you have finished editing the details about a node.

Whether you add or edit a node, this will launch the node editor, which has the following features:

* The first (and default) mode is in WYSIWYG[[1]](#footnote-1) editor that provides an interface somewhat akin to a word processor. In OpenLabyrinth this is provided by the TinyMCE component[[2]](#footnote-2). There is a button within the editor panel [HTML], which changes you to a direct editing mode, where text and HTML code is entered directly into the content and info fields. This is sometimes useful for fixing small formatting glitches or for inserting other more sophisticated HTML code into the page. Other icon buttons within the TinyMCE editor panel allow you to change text to Bold, Italics, Underline etc, as well as a number of other editing features.   
    
  

Figure 15: the WYSIWYG tool bar

* Node title: this is both the title displayed in the lists of nodes and at the top of the page and the default text of any link to the node (although this can be overridden – see section 5.5).
* Node content: This is what is shown to you and is where narrative, images and instructions are added. You can also insert links to Media Resources, Questions, Avatars, Elements and InfoButtons in here using a simple wiki-style of notation. See section 6.7 for more information on how to insert these references. Note that to add a Media Resource such as an image you need first to upload the file and then copy and paste its wiki-style reference into the node content box where you want the file to appear (see files section).
* Supporting information: additional content can be added in this field. Any content in the info box for a node will be displayed when you click the InfoButton at runtime. This launches a popup window containing the supporting information. This would typically be used to provide tips, commentary or advice from a character such as a tutor, or links to other materials and references outside OpenLabyrinth.
* Supporting information keyword: this is the name of the wiki-style link that must be inserted into the Node Content window at the point where you want the InfoButton icon to appear in the text. If you do not insert this keyword link then no InfoButton is shown. (This has changed from OpenLabyrinth v2, where you could only have the Supporting Information displayed after all the other node content.)
* Counter functions: for every counter created, this is where they can be dynamically changed. You can leave this blank or use one of ‘+’ or ‘-‘ or ‘=’ plus an integer. For instance ‘+5’ adds 5 to the current counter while ‘=6’ sets it to 6 no matter what it was before. See Counter Grid section for more information on setting counter scores.
* Exit node probability settings: this toggles on off whether just one of the linked nodes will be presented at random or all of them presented at once.
* Node Conditional settings: this allows you to set conditional rules for entering the current node when playing the case. These rules specify what nodes need to have been visited before the current one can be accessed. When creating a conditional rule, a message to you is specified along with a Boolean statement of which nodes are required. For instance {1}AND({2}OR{3}) requires you to have visited node 1 and either nodes 2 or 3 before accessing the current node. Editing a conditional rule deletes any previous rule requiring it to be reset.
* Link presentation style: this changes how the linked options are displayed to you:
  + text (default) – each choice is shown as a plain text hyperlink.
  + dropdown – each choice is an option in a drop down list.
  + dropdown + confidence – each choice is an option in a drop down list with a second drop down to let you select how confident they were at making the selection.
  + type in text – user types in what they think the option should be. If, after the first 3 letters are entered, this matches one of the available choices then that choice is selected.
* Node priority: this flags whether the current node must be avoided or must be visited – this is only used in the session reporting and feedback. See section 7 on [Feedback and Reporting](#_Feedback_and_Reporting_1). It does not affect the user’s navigation of a case. This is not the same as ‘Conditional Rules for node’ above
* Set as Root button: this sets the current node as the root (starting point) node for the labyrinth.
* Undo links: this allows users to undo a node they have visited by removing it from the session track. In any given node scroll down to “enable undo links” and select whether this feature should be on or off for this node. This feature is not yet properly implemented in OpenLabyrinth v3.
* Link to end and report from this node: this is off by default. If turned on, a link is added to end the current session and get feedback – see [Feedback and Reporting](#_Feedback_and_Reporting_1) section for more on this. Note that only logged in users can view such Reports.
* Remember to click the ‘Save changes’ button at the bottom of the page, once you have finished editing the details about a node.

## Links

The links editor in a map lists every node in the left hand column and every link from that node in the right. Every left hand node entry allows you to edit its particular links or preview it. Every linked node entry on the right allows you to go to that particular node’s entry in the left hand column or preview it.

The editor for a specific node’s links allows you to:

* Change the type of linking from the current node: there are three kinds of node linking that can be set here:
  + Ordered: this allows you to specify the order that links will be shown. This ordering is set in the current link editor using numeric dropdowns per link – they evaluate low-high.
  + Random order: this means that the order of the available links from the current node will be randomised every time the node is loaded.
  + Random select one: this means that just one of the available links will be randomly selected – note that this property can also be set in the node editor.
* Edit current links from the current node: this allows you to order links (if linking type has been set to ‘ordered’), delete links or edit them. If you are editing a link the form looks very similar to that for adding new links.
* Add a new link to the current node: this allows you to select a node to link to from a dropdown list of every node in the current map not already linked from the current node. You can also (optionally) set alternative text for the link (the default is the target node’s title) or set a path for an image or icon to be used instead of a textual link.
* Add a counter function to a link – this will only activate when the link is clicked. This is different from the previous OL model of counter functions only being available on the node.

## Play

This starts the labyrinth running in a new window. You can also start Play from a particular node by clicking on its name in the Nodes editor listing. If the map has not already been played in this authoring session, you may get an error from the Kohana database. We expect that this wee bug is fixed soon.

## Using the Visual Editor

The Visual Editor is a graphical tool that allows you to add, edit, move and delete nodes, links and node content rather like we previously did with the VUE concept mapping tool, but now inline to OpenLabyrinth so that edits are made directly to the labyrinth and can be previewed at once.

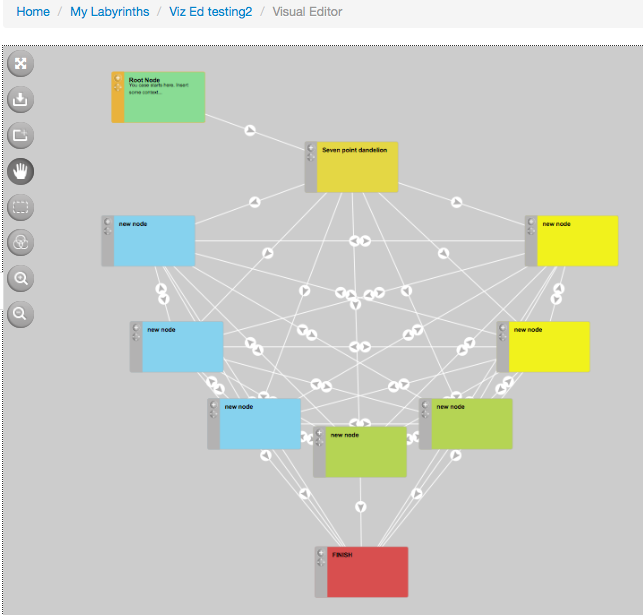


Figure 16: the Visual Editor

|  |  |
| --- | --- |
|  | zooming: click on the Plus or Minus buttons to zoom in or out - the default is 100%. Quick keys are Alt-plus and Alt-minus. Avoid the use of Ctrl-Plus or Cmd-Plus. This can produce unexpected effects. |
|  | panning (moving around): click and drag in an open area to drag the workspace around - a newly imported labyrinth may need to be dragged into view - first zoom out to find it, drag to the centre and then zoom in. |
|  | saving your work: the visual editor needs to be updated to save changes to the database. Quick-key is Alt-S. |
|  | moving node boxes: click and drag in the left side bar in a node box to move it around the screen. All links to or from the node box will move at the same time. |
|  | editing text in node boxes: click on the title or body text of a node box to start editing it. A popup floating editor panel will allow you to change text, color, title, info, and a number of other node attributes. Remember to Save before moving to the next node or link. You can leave this panel open and it will refresh with the newly selected item’s contents. You can drag the panel around the desktop to keep it handy. |
|  | adding node boxes: click on the + icon at the top left of a node box to add a new node linked from the one you clicked. Alternatively click the + icon at the top left of the editor to create a new node not linked to another node. Note that the viewer needs to be updated before any changes are added to the database. |
|  | Full screen mode: minimize other OpenLabyrinth elements to maximize the drawing desktop space. |
| mvstack5 | adding links: click on the target icon on the left hand side of a nodebox to create a new link. Click and drag the target on to the node to be linked and let go. A new link has been created. Note that the viewer needs to be updated before any changes are added to the database. |
|  | Multi-select tool: you can select many nodes at once. Draw a box around them and then move/copy/cut/paste them en masse. Quick keys for Copy/Cut/Paste are Ctrl-C, Ctrl-X, Ctrl-V (not Cmd-C etc) |
|  | Insert pre-template: you can insert a sub-set of nodes. Select from various pattern designs: linear, branched, dandelion. |
|  | editing links: click on the arrow icon in the middle of a link to edit it. Select one of ‘Direct', ‘Back', ‘Dual' or ‘Delete' and then click Apply. Note that also add a new Link Label here. This will be the text in the choice provided to you, instead of the title of the target node |
|  | change a node box background colour: click on a node box then select the Actions tab in the floating panel to change the background to selected colour from the colour wheel. |

Please also note the following:

* From version (2.5) VUE map imports include the X, Y and RGB properties of every node. Previous imports and any other node content will not by default have these properties and will be shown in a grid format.
* Updating the viewer will save any changes. It is recommended that this is done regularly, particularly when creating new nodes to ensure they are saved and managed properly.

## Finishing a case

If you are logged in, on the OpenLabyrinth home page there is a ‘My OpenLabyrinths’ link from which you can view a list of every labyrinth you have started, behind which there is a user session report that shows what path was taken, how much time elapsed between entering and leaving the node and the varying level of any counters. A histogram of this data is also available. See section 7 for more information on [Feedback and Reporting](#_Feedback_and_Reporting_1).

# Advanced Authors

* 1. **User Interface**

There may be a number of different elements on display (see figure 17) including:

* Title: every node has a title, which is typically displayed at the top of the page.
* Node content: although not mandatory almost all labyrinth nodes will have some text content for the user. Typically that will describe the consequences of having made the previous choices of paths and possibly things to consider in making the next choice.
* Linked options: the way to navigate an OpenLabyrinth case is to click on one of the available options here. These will usually show the title of the node you are linking to.
* Review your pathway: a clickable track of your pathway through the labyrinth since you started is available. Click on a link to go back to that point – note that this does not roll back the track of which nodes have been visited. Every action is recorded in a session. This may also affect how some Counters and Rules operate.
* Counters: a simple way to show your progress in a game. Authors can show or hide counters on each node, minimizing distracting information when not needed. There may also be a display of how the counter’s value has changed since the last node.
* Map and Node ID: this shows the current labyrinth and node ID number. If there is a problem with a case, the case authors will appreciate you making a note of which node ID number caused your problems.
* Reset link: this ends your present session and restarts you in a new session within the same OpenLabyrinth case.
* A link to OpenLabyrinth home page: clicking the OpenLabyrinth icon returns you to the home page.
* Other graphics, links, tools and text may also be displayed depending on the current skin.

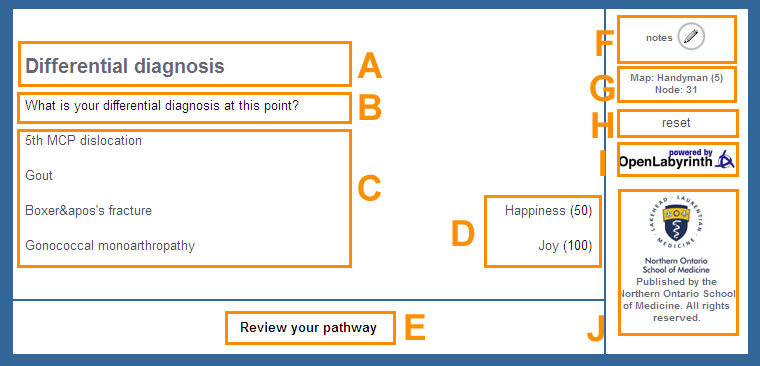


Figure 17: a typical labyrinth screen’s elements:

*A: Node title; B: Node message/content; C: Links/options; D: Counters – label plus current value; E: Link to review nodes viewed in the current session; F: Open note taker; G: Current labyrinth and node information; H: Reset/restart current labyrinth; I: link to OpenLabyrinth home page; J: Skin-specific graphics and text.*

## Node Grid

Sometimes you just want to edit the content of all of the nodes in one go. That is when you use ‘Node Grid’. Basically it provides a basic text box editor for the title and body of every node in a labyrinth for every node in a labyrinth. Be aware that you are working with the whole case at once. This is important when you are part of a team that is collaborating on a case. See [Collaborative editing as a team](#_Collaborative_editing_as).

## Sections

Node sections are a way of organising nodes into logical groups. They can be used to help authoring, particularly where there are a great many nodes or several nodes have the same title. They can also be used to create a menu to help users browse through a labyrinth.

* You can add and edit node sections and assign nodes to node sections from ‘List/edit this labyrinth’s node sections’ link on the main editor page.
* You can add a node section by typing a label into the form at the bottom of the node sections page and clicking submit.
* You can edit an existing node section by clicking on the ‘edit’ link next to its name on the node sections page (which also lists the nodes assigned to each section). This provides a list of the current nodes in the section that allows you to order them and remove them from the current section. You can also add any unassigned node to the current section. Note that a node can only be assigned to one section at a time.
* You can control how the node sections are shown to you. The choices are: don't show, visible, navigable (clickable). In the latter the section label becomes a hyperlink to the first node in the section.

## Feedback

This lets you control how the feedback will be presented to the user at the end of the session. See [Feedback and Reporting](#_Feedback_and_Reporting_2).

This part needs a bit of work so this section of the user guide will be updated at that time.

## Questions

You can now create several kinds of questions:

* Free text questions have no right or wrong answer. OpenLabyrinth stores but does not evaluate the text. The appearance of the question in terms of the size of the box can also be set.
* Multiple-choice questions can have right, wrong or neutral answers. Only one choice is valid (sometimes known as radio buttons). The score from any given question can be used to control a Counter, which can in turn be used to control game play within a labyrinth.
* Pick choice question - can have multiple valid selections (sometimes known as check boxes). Each response can also control a Counter.

Multiple choice and pick choice options can be stacked vertically or horizontally across the page. Be careful with horizontal stacking because the response feedback is displayed by the choice, which can spoil your layout.

These are created in the ‘Questions’ part of a labyrinth’s editor and, once created, a [[QU:xxx]] tag is created. Copy and paste this into a Nodes Content box to render the question at runtime. Multiple questions can be placed on a Node page.

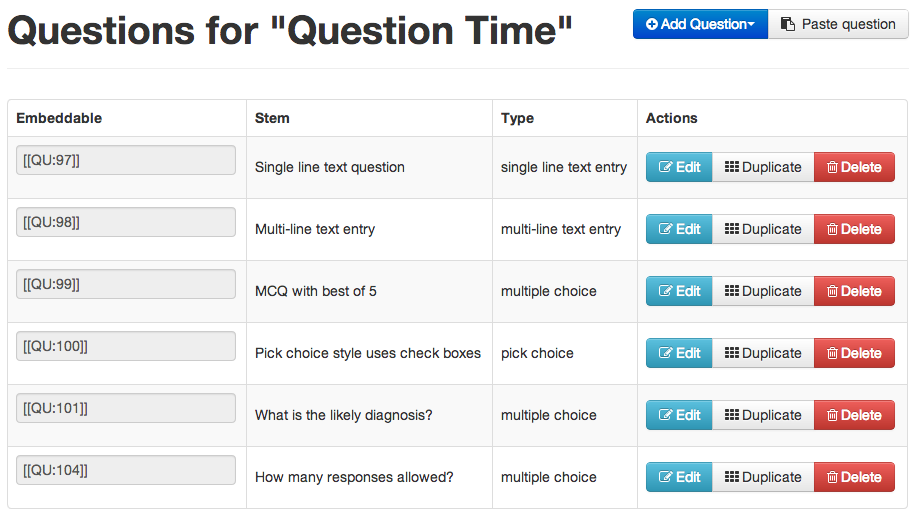


Figure 18: the Questions editor

After clicking on the blue ‘Add Question’ button and choosing which style of Question to create, you will then see a page like this:

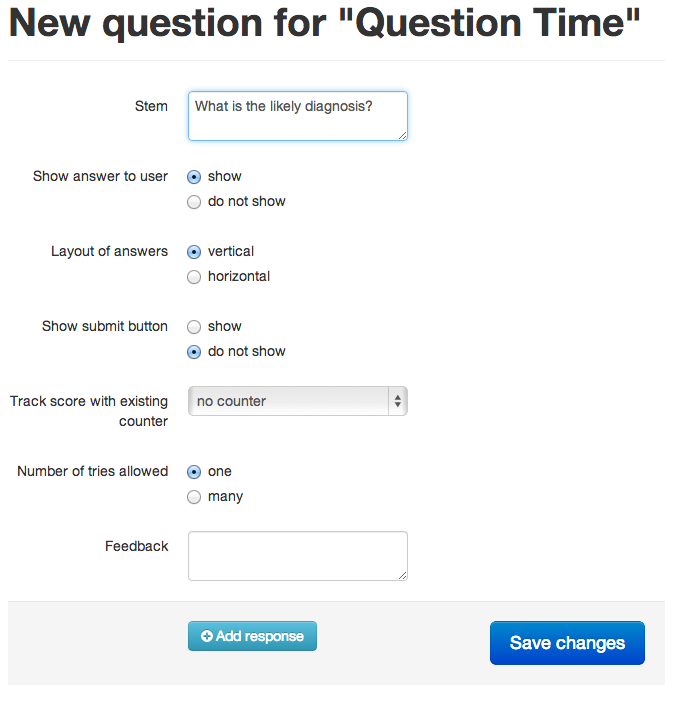


Figure 19: starting off in creating a Question

The fields are pretty self-explanatory. The additional Submit button option is a way to force the user to exit to a particular node, perhaps to show Counter scores or provide specific instructions. If there is no Submit button, the Question’s responses are evaluated when the user leaves this nodes and moves on to another.

You can add as many responses as you wish. (Previously in OpenLabyrinth v2, you were limited to 2,3, 5 or 9 part questions.)

If you have more than one Question per page, all of the responses selected will affect the Counter scores separately. Note that the answers for all questions are provided in the end of session Report.

For some authors, we know that using Questions will be popular and that oftentimes, the Questions will only be slightly different from one situation to another. To decrease the tediousness of recreating many similar Questions, we have provided two methods. Within the same case, you can simply Duplicate a Question and then edit the duplicate question, changing responses or counters as desired. If you want to copy/paste a Question from one labyrinth to another, it is only slightly more involved. First make a note of the Question ID number that you wish to use for your source example. So, if you want to use [[QU:1234]] in another labyrinth, do not simply paste that wiki-style link into that labyrinth like you would with an image [[MR:23]] – this will not work. Go to the destination labyrinth where you want to insert the question and in the Questions editor page, click on the grey ‘Paste question’ button beside the blue ‘Add question’ button. This will pop up a small dialog like this:

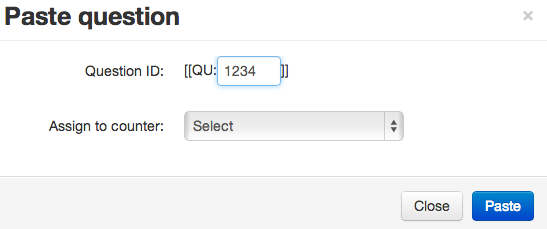


Figure 20: inserting a Question into another labyrinth

Simply enter the number of the Question (you do not need the wiki style square brackets surrounding the number – these are attached for you). If you want to assign the scores to a Counter, select one here. Click the blue ‘Paste’ button to insert a copy of the original Question into the current labyrinth. You can edit this Question in the usual manner thereafter.

It is important to understand that the free text questions only provide simple recording of text input. OpenLabyrinth does not try to interpret the answers. Natural language processing is difficult to get right and easy to get wrong, leading to frustration for both players and authors.

We are working on variations in ‘limited scope text matching’ as a means to provide increased flexibility for case authors who do not wish users’ answers to be completely pre-scripted. For more information on this, consult [Key Feature Problems and Matching](#_Key_Feature_Problems).

## Avatars

Avatars are animated simple images of the characters you might want to include in your labyrinth either as passive representations of the characters in your narrative or taking part via speech or thought bubbles. There can be any number of avatars per labyrinth – for instance the same character in different settings and saying or thinking different things or perhaps many different interacting characters.

Each avatar can be customized as follows: Sex, age, eyes open or closed, outfit, mouth shape, the colour of their outfit (for some outfits only), their nose type, hair type and colour, three different layers of accessories such as wounds, glasses, bandages etc, skin tone and colour, background colour, background scenery, background weather and the kind of speech or thought bubble used and what text should go in it.

Once created each avatar has a reference that looks like [[AV:123]]. Copying this tag and pasting it into the contents of a node or info box and allows it to be incorporated and reused anywhere in the current Labyrinth. You can duplicate an existing avatar for instance to create a series around a single character, and you can edit them and delete them. See figure 21.

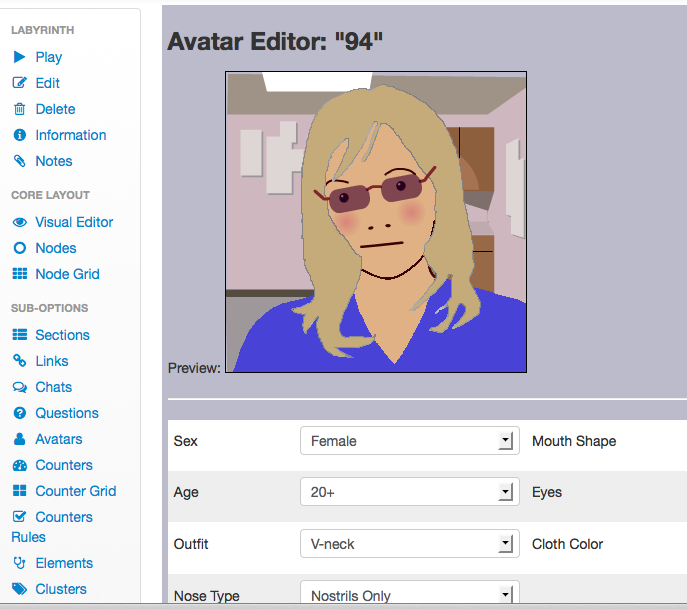


Figure 21: the avatar editing screen

In OpenLabyrinth v3, these are created using HTML5. We decided to stop using Adobe Flash because a number of major software vendors like Apple decided to stop supporting it on their devices. Apple iOS mobile devices do not support Flash. Frustratingly, Microsoft has been poor at adopting HTML5 – neither IE7 nor IE8 supports HTML5 properly. However, Windows users have an easy workaround and can install a much better browser for free. We recommend the use of Firefox or Chrome for running our cases.

## Files

You may wish to add images, sounds, videos, documents, spreadsheets or other discrete files to your OpenLabyrinth map. To manage your files click on ‘Files’ link in the main menu. This launches the file manager window (see figure 22).

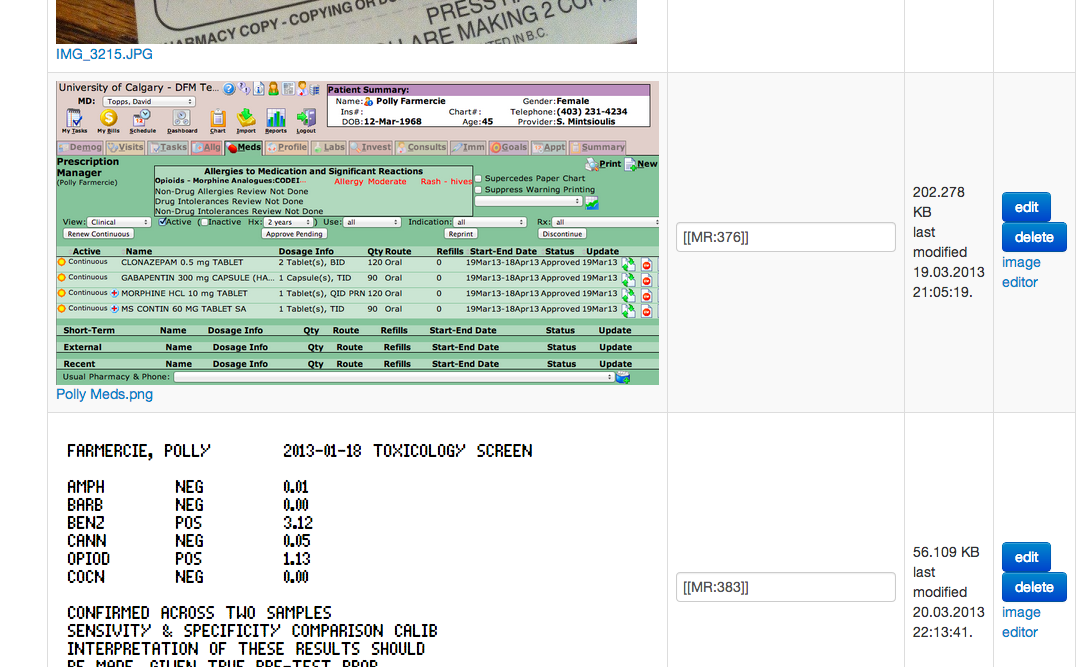


Figure 22: map files editor.

For each file uploaded into the current labyrinth you have:

* A wiki-style file reference: this takes the form of [[MR:xxx]] where MR indicates that this is a media resource and xxx is the unique file ID assigned to the file on upload. By pasting the reference into a node content box the file will be displayed at runtime.
* A resource preview: this just shows what the resource looks like (images only).
* A metadata view and editor: this shows basic metadata for the resource, allows you to edit the metadata and it allows you to delete the file from the current labyrinth.

In addition you can upload a new file by choosing it and clicking the upload button (see Table 2 for list of supported file types’). Once a file has been uploaded it should be edited for its metadata (this will allow for both better control of the resource as well as registration within the Media Resource MVP service – see section on ‘Files’). Note that you should always ensure that any material you use (both text and images) is not covered by any copyright or consent restrictions. Liability for inappropriately used materials rests with the author.

|  |  |  |  |
| --- | --- | --- | --- |
| File Type | Display | Extension | Notes |
| JPEG image | Inline | .jpg, .jpeg | JPEGs and GIFs are the only supported image formats in OpenLabyrinth |
| GIF image | Inline | .gif |
| Acrobat PDF | Link | .pdf | PDFs may be created from many applications including Adobe Acrobat |
| Shockwave Flash | Embed | .swf | Runtime media files from Adobe Flash |
| Microsoft Word | Link | .doc, .docx | Standard document formats |
| Microsoft Excel | Link | .xls, .xlsx |
| Microsoft PowerPoint | Link | .ppt, .pptx |
| Rich Text Format | Link | .rtf |
| QuickTime video | Embed | .mov | Video formats |
| MPEG-4 video | Embed | .mp4 |
| Windows Media | Embed | .wmv |
| Real Stream (RAM) | Embed | .ram |
| Real Stream (RPM) | Embed | .rpm |
| Flash video | Embed | .flv |
| MP3 audio | Embed | .mp3 | Audio formats |
| WAV audio | Embed | .wav |
| AAC (m4a) audio | Embed | .m4a |

Table 2: file formats supported by OpenLabyrinth

## Counters

Counters are comprised of a label and a value, which can be dynamically changed as you works through a labyrinth. There can be any number of counters and each of these counters can have any number of rules that can be triggered by a counter’s value. Values can now be integers, floating point or strings.

Counters need to be created globally for a map. To create a new counter or edit an existing one click the ‘Counters’ link. This opens the counters editor, which lists each existing counter with links to edit, preview or delete it along with a link to create a new counter. Adding or editing a counter opens the counter editor window (see figure 28).

The counter editor allows you to define/change:

* Counter title: this is the label shown for the counter.
* Counter description: an optional text description of what the counter is and what its purpose is.
* Path for a counter icon: an optional path for a graphical icon for this counter.
* The starting value for the counter

Once created, each counter will be displayed (except on some skins) to you indicating its title and current value. If the value is changed at a particular node that value change is also displayed. Clicking on a counter link will launch a popup window with full details about that counter. Changing the value of a counter as you moves through a map is set in the node editor which has a function box per counter: you can leave this blank or use one of ‘+’ or ‘-‘ or ‘=’ plus an integer. For instance ‘+5’ adds 5 to the current counter while ‘=6’ sets it to 6 no matter what it was before.

## Counter Grid

Counter Grid shows a function box for every node and every counter so that counter functions can be set all in one go. Counter visibility can also be set on or off, for each node in the grid, or all on, or all off.

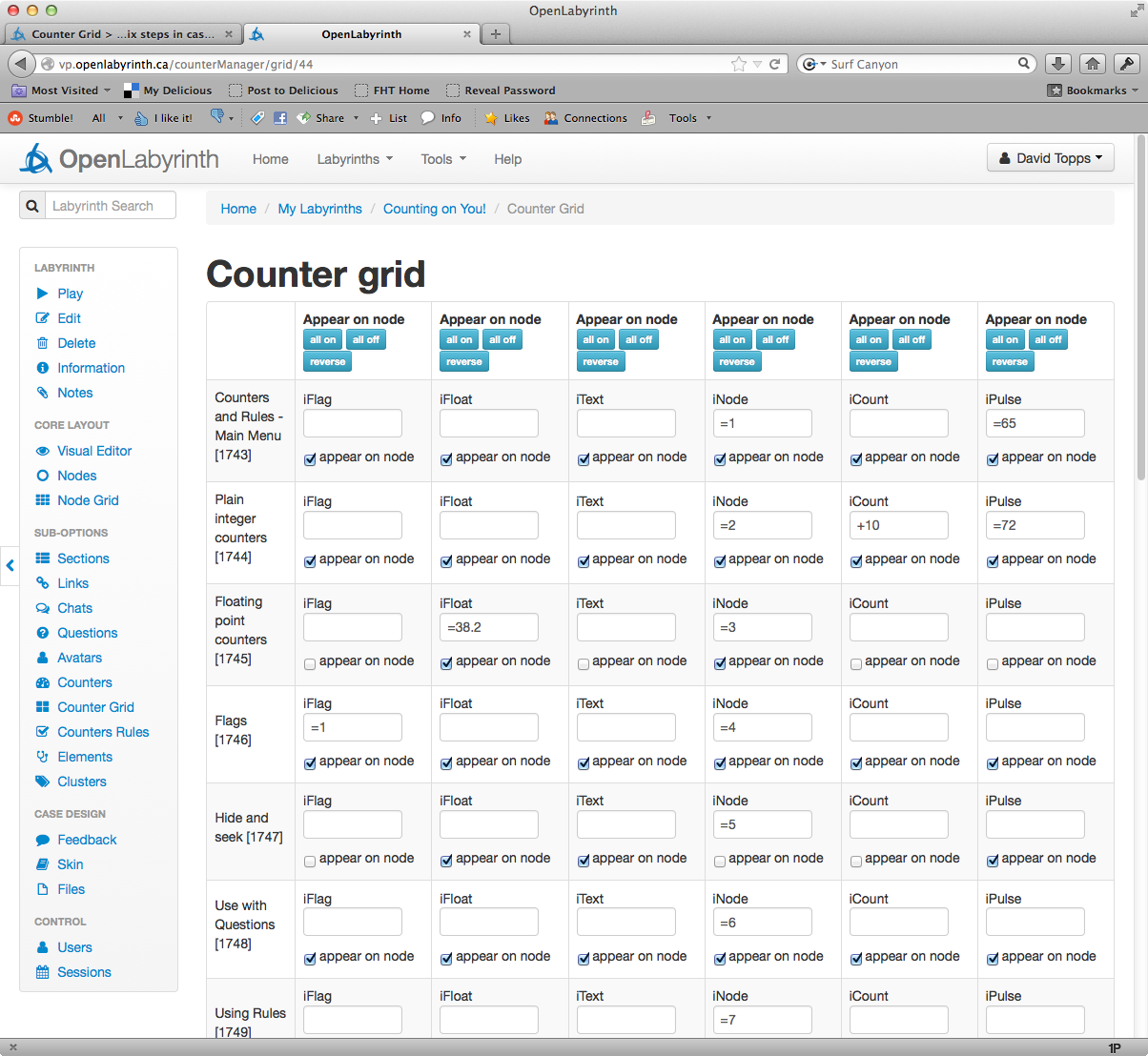


Figure 23: Counter Grid with six Counters

Be aware that this tool affects the case widely. This is important if you are collaborating with team members when writing a case. See [Collaborative editing as a team](#_Collaborative_editing_as_1).

## Counter Rules

Counter rules can be added from the editor for a particular counter. Each rule consists of the following components:

* An integer value
* An operator: equal to, not equal to, less than or equal to, less than, greater than or equal to, greater than
* An action if the condition is met:
  + Go to one of the nodes in the current labyrinth Note that the counter value should be reset at the target node or the program will loop
  + Show a message in place of the default node message

For example: if counter 3 is greater than or equal to 50 then show message “well done”.

From this versions counter functions can also be set on links. This means that the value of a counter can be changed based in the link selected rather than the node entered.

From this version the visibility of each counter can be controlled by setting it to ‘show’ or ‘do not show’.

For more detailed information, see [Syntax for Counter Rules](#_Syntax_for_Counter_1).

## Data Elements

The MVP model is made up of four components:

* VPD: virtual patient data – functioning as the electronic patient record, a VPD is a collection of VPD elements. Although designed to support clinical data this could just as easily be used to support any reusable data element such as a character name or unit of measure. These are therefore called ‘data elements’ in OpenLabyrinth.
* MR: media resources – this includes all supporting files including images and documents. An MR Set is a collection of MR elements. Each MR element has properties of name, path, mime type and arguments. Media resources are handled using the OpenLabyrinth file manager.
* AM: activity model – this describes what you can (and cannot) do. Activities are described as a series of nodes containing narrative and educational content with links between them. Both nodes and links have rules associated with them to enhance the options for educational game play. This maps directly to the node and link aspects of OpenLabyrinth.
* DAM: data availability model – aggregations of VPD and MR elements. DAM contains at least one VPD element or MR element, although more typically it contains many such elements.

In terms of OpenLabyrinth, its activities correspond to the OpenLabyrinth activity itself, MR equates to the file handling capacity of OpenLabyrinth while VPD and DAM functionality have been added as ‘data elements’ and ‘data clusters’ respectively. VPD and MR data elements can be linked directly into an activity node or via a DAM aggregate of VPD and MR elements (see figure 24).

OpenLabyrinth can both export MedBiquitous Virtual Patient packages (a zip file with XML and other files inside) from a labyrinth and it can import MedBiquitous Virtual Patient packages to create new labyrinths.

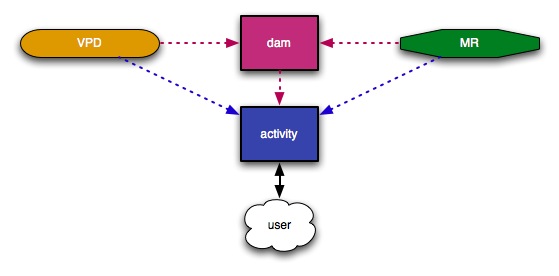
**

Figure 24: The relationship between VPS, MR, DAM and activity elements in OpenLabyrinth.

*The user only works with the activity; all elements are therefore surfaced within the activity.*

OpenLabyrinth supports the MedBiquitous Virtual Patient (MVP) data specification, a key aspect of which is the use of data elements. OpenLabyrinth supports the following MVP element types, each of which has a different structure:

* VPDText
* PatientDemographics
* AuthorDiagnoses
* MedicationInterviewItem
* PhysicalExam
* DiagnosticTest
* DifferentialDiagnosis
* Intervention

To use OpenLabyrinth data elements:

1. Select ‘data elements’ from the editor menu. This will give a list of all the current data elements and a link to create a new element
2. You can edit or delete any of the data elements
3. On the new element page select the type of element you want to create from the drop down menu – this will load a form to be completed to create the element

Using a data element in a labyrinth involves pasting its tag into the content for any given node. The tag has the format of ‘[[VPD:’ then the element’s ID and then ‘]]’, i.e. [[VPD:123]] – see Figure 25. You can also use data elements in data clusters.

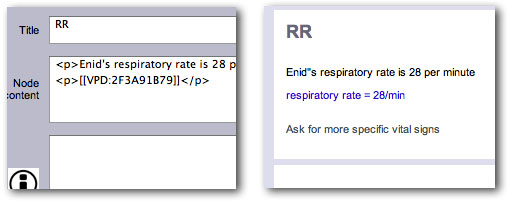


Figure 25 an embedded data element in the node editor (left) and how it renders on screen (right)

## Data Clusters

Integrating elements – there are two ways of linking in VPD or MR elements to an activity node: as individually embedded elements or as a collection of elements via a data cluster.

Click ‘clusters’ on the editor menu – this lists all of the current clusters. You can edit the contents of a cluster or delete a cluster or you can create a new cluster. Within a cluster you can add or remove data elements or media elements (such as pictures) and change the order in which they’re shown to you.

## Session reports

This lets you view all of the sessions run on the current labyrinth. See [Feedback and Reporting](#_Feedback_and_Reporting_3).

## Users

A map’s authors are the only ones who can edit it and if the map’s security is set to private they are also the only ones who can see it. By default the user who created a map (by any of the creation methods) is an author on the new map. Additional authors can be added (or removed) using the map’s author editor which is accessed by clicking on the ‘Tools | Manage Users & Groups’ link in the main menu. This provides a list of all the registered OpenLabyrinth authors that can be selected to add the user as an author of the current map. Delete them to remove them from the current map. Note that you cannot add or remove yourself as an author.

Learners can also be added to a map so they can access it (but not edit it).

OpenLabyrinth also supports LDAP authentication. Your system administrator can establish links from the OpenLDAP interface to your local LDAP server. This allows you to control larger groups through your normal access control lists and service management tools. See [Authentication systems](#_Authentication_systems) for more information about this.

## Export

There is now only one kind of export – MedBiquitous VP:

* You can export a labyrinth to the ANSI/MedBiquitous Virtual Patient standard package format – see section [Creating a OpenLabyrinth Map by Importing a MedBiquitous Virtual patient Package](#_Creating_a_OpenLabyrinth) for more details. To create an MVP package click on ‘export MVP’ from the editor main menu, select the appropriate licence to release the package and then create and download the zip file.

## Duplicate

Any labyrinth you have edit access to can be duplicated as a template for a new labyrinth by clicking its ‘duplicate’ link on the editor page. A basic copy of the original labyrinth is created (called “Copy of …”) allowing you to change any aspect while keeping the original untouched. Note that duplicating just takes the basic structure and does not import files, rules or other additional properties.

## Delete

You may wish to delete a labyrinth you have started. To do so click the ‘Delete’ link on the main menu – you will be challenged whether you really want to delete this map. If you are sure that you do then click the ‘go ahead and remove this labyrinth’ button. Recognising that some people may wish to resurrect deleted maps, they are not permanently deleted but disabled and held offline. Contact an OpenLabyrinth administrator to reinstate a deleted map or go to the database and change the mapEnabled flag to ‘y’ in the MAP table.

## Author Notes

A simple note taker can be run for the authors of a labyrinth. Click on the pencil icon to load the author notes editor. This can be viewed and changed by any of the editors of any given labyrinth. This will be enhanced in future versions by semantic linking to external metadata tables, SPARQL end points and IMS-LTI consumers.

notes

## Key Feature Problems and Matching

If the labyrinth type is set to Key Feature Problem then any node within that labyrinth can be set to be a matching question rather than providing links. Turn matching on by going to any node in the current KFP typed labyrinth and scroll down to “add text matching”, turn it on and submit the page. Go back and re-enter the node edit page and there is now an ‘edit button’ next to the “add text matching” buttons. Click edit to start adding different strings to be matched (up to 12 per question as well as the score for a match and whether the match is critical or not.

You can also set the following:

* counter controlled – this is the counter that is changed as a result of matching and adds the associated score to this counter
* strict [off][on] – this changes whether a string must be matched strictly or not – for instance if the match term is “cardiac arrest” if you enters “cardiac” they will get a match only if strict is turned off.
* number of options for user – this is the number of text boxes and therefore tries the user has to submit candidate matches
* next node – this is the node that you is taken to after the matching has been processed

In a matching-enabled node you is presented with a series of text boxes and a submit button – see figure 26.

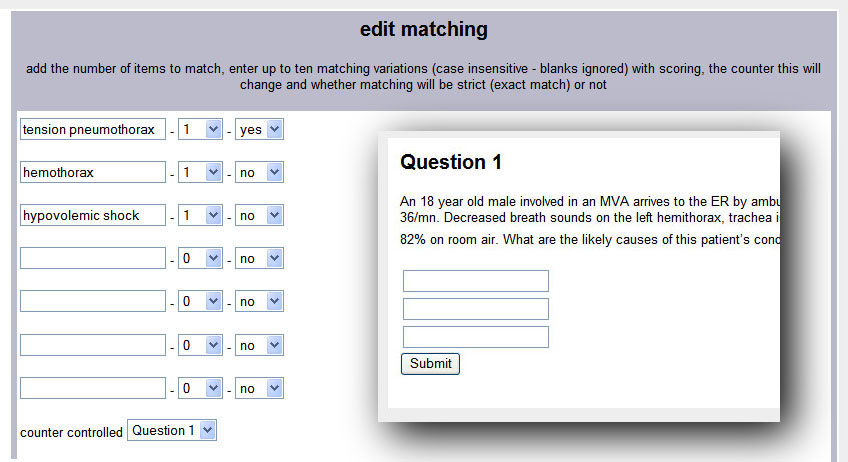


Figure 26: the matching editor and the way a matching question is presented to you

(in this case with three attempts)

# Feedback and Reporting

As with any educational program a labyrinth is at its most useful when it can inform a learner how well, or how badly, they performed. It is even more useful when accompanied by suggestions about how they might improve their performance in future.

## Session Reports

As has already been mentioned each labyrinth user session is tracked and is available to its authors as a report. A user session is started each time you go to a map’s root node. The tracking involves recording each node selection along with the time and current score and counter values at that point. The reports are available in one of two ways:

* Users can see reports for every OpenLabyrinth they have run – this is in the ‘My OpenLabyrinths’ section linked from the home page
* Authors can see reports for every session within a particular map – this is in the ‘user session report’ section linked from the main map editor.

Each report consists of a list of nodes visited (representing the user’s path through the map) along with the current score and how long they spent at the node (in seconds).

This allows a histogram to be drawn representing the session to be drawn up (see Figure 27). The value of each counter is also recorded and is available as a plot (see figure 28).

The data can be downloaded as a CSV file for further analysis.

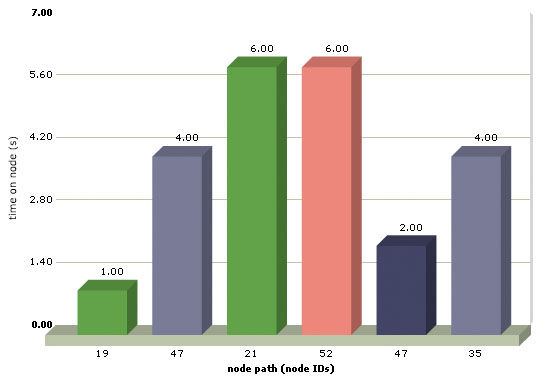


Figure 27: a typical OpenLabyrinth report histogram.

In this session you spent 6 seconds on nodes 21 and 52. Node 21 is a ‘must visit’ node (in green) and 52 is a ‘must avoid node’ (in red).

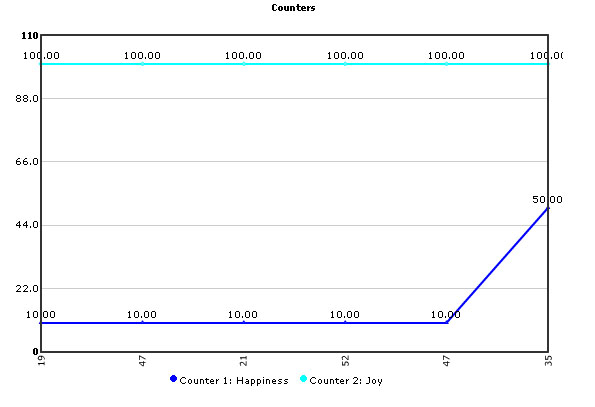


Figure 28: a typical counter trace

There are two counters here ‘happiness’ which started at 10 and rose to 50 on the last node and ‘happiness’ which remained at 100 throughout

## Feedback Report

You can also be given a feedback report. To do this first enable the node property of ‘link to end and report from this node’ for any node from which you want to allow the user to get their report. When you click the report link you get the following:

* Metadata such as the user ID, the session ID and the labyrinth name.
* The start time and the time taken to complete
* The total number of nodes visited as well as the number of ‘must visit’ and ‘must avoid’ nodes visited
* General feedback irrespective of what you have done
* Feedback in response to specific nodes visited
* Feedback in response to the number of ‘must visit’ and ‘must avoid’ nodes visited
* Feedback based on the time taken to complete
* Feedback: on the final values of any of the current labyrinth’s counters
* The list of nodes visited
* The histogram of time spent per node
* The graph of counter values through the session

## Feedback Options

Feedback is an essential part of any educational activity. Although the content of each node is a form of feedback about the choices the learner made OpenLabyrinth also supports summative feedback that is only provided once the session is complete.

Each labyrinth can be set to provide feedback on user performance on the following areas:

* General feedback to you – for instance “That was a hard problem …”
* Feedback in response to specific nodes visited – for instance “You chose to get an MRI, that was an expensive option given the circumstances …”
* Feedback in response to the number of ‘must visit’ and ‘must avoid’ nodes visited – for instance “You made 4 choices that should have been avoided …”
* Feedback based on the time taken to complete – for instance “You completed the task in less than 2 minutes, were you really thinking about what you were doing?”
* Feedback: on the final values of any of the counters – for instance “Your morale slipped below the critical point …”

User feedback can be configured using the ‘feedback’ option in the editor menu.

# Global Functions

In addition to the creation and running of OpenLabyrinth maps there are a number of OpenLabyrinth administration services including user management, reporting and import/export:

## Users and Groups

All authors need to have a login to OpenLabyrinth. OpenLabyrinth super users have the ability to create OpenLabyrinth accounts using the link from the home page. Each account requires a username, password, full name and email address. This is where the user interface language is set for a given user. A user may also be given super-user access.

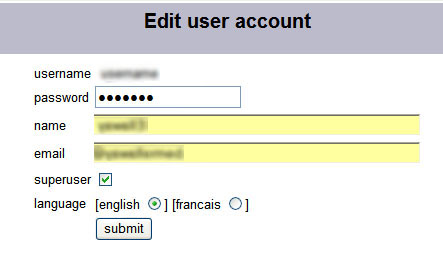


Figure 29: the add user screen

Groups are collections of users. The tools for creating groups and managing their membership are below those for users on the same page.

## Language support

The user interface language is set in the user account creation or editing pages. If you change your own user language then you must logout and log back in for the changes to take effect. Currently there are just two supported languages – English and French. Note that the user guide is currently available only in English although a French translation would be most welcome. Note that the user interface phrases are stored in the “interface.xml” file sitting in the documents folder at the OpenLabyrinth root.

At present the French translation is likely to be pretty poor limited as it is to the author’s memories of school French and the quality of Google Translate. Corrections and improvements are invited. Furthermore anyone willing to translate the 370 or so interface elements into a third or fourth language are welcomed and encouraged. Simply add a language tag at the top of the XML file – eg <language ID=”KN” name=”klingon” /> - and then add an additional element for each phrase – eg <phraseKN>qapla</phraseKN>. Return to the author for inclusion in the next release.

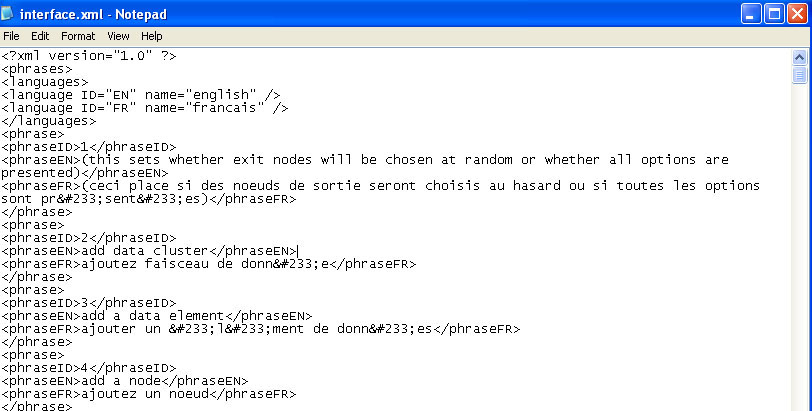


Figure 30: the interface.xml file that can be found in documents/interface.xml

## Presentations

Groups of Labyrinths can be collected together as ‘presentations’ for instance as a course page or a bank of assessment items. To create a presentation click on the ‘presentations’ link on the home page, add the details for the presentation such as any text to be shown and user options and then submit. Edit a presentation to add specific labyrinths to it. Note that you must be an author on any labyrinth you want to include in a presentation – see figure 31.

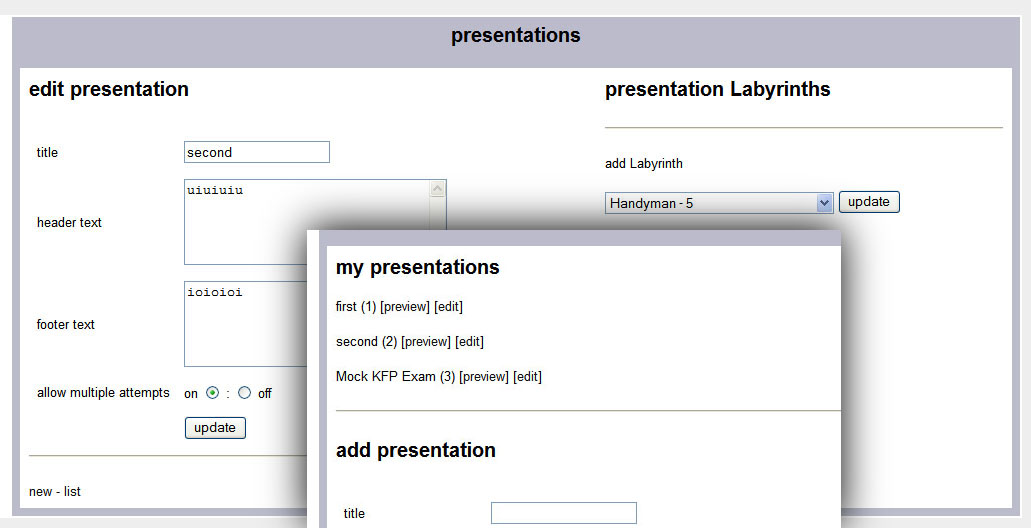


Figure 31: the Presentations authoring screens

## Collections

This allows you to create a collection of different labyrinths. For instance, there may be a set on a particular topic or that use a particular approach. A labyrinth can be allocated to multiple collections.

## MedBiquitous Virtual Patient Export

OpenLabyrinth currently exports to the MedBiquitous Virtual Patient XML format. To create a package, click the menu item Labyrinths | Export Medbiquitous. This will run a script that creates a new folder, copies the required files into it, writes XML files for the various required components and then creates a zipped archive of the folder and its contents – the user is then presented with a link to the zip archive.

The key elements are labelled as follows:

A The Zipped package

B The IMS Manifest XML file that lists the contents of the archive

C The metadata XML file in LOM format

D Activity Model XML file

E Folder containing all of the media objects

F Virtual Patient Data XML file

G DAM XML file

The other files and folders are required objects for a SCORM package.

## MedBiquitous Virtual Patient Import

In a similar but reverse way to MVP export, OpenLabyrinth can create a new labyrinth by importing and MVP package. To create a labyrinth, click the menu item Labyrinths | Import Medbiquitous. This will run a script to upload an MVP package.

You select the zip archive, this is uploaded to a new temporary directory, unzipped, the XML files parsed and the data written into the database, files copied to new directories and when all the importing has been done the temporary directory deleted and a link to the new labyrinth presented to the user.

This works best with MVP packages created by OpenLabyrinth v2.6.1 or later. Prior versions miss out quite important detail in their packages.

# OpenLabyrinth Remote Services

[ NOTE: the OpenLabyrinth services are being significantly expanded – see separate release for code and details of how to use these new functions. The text in this section has been copied from the OpenLabyrinth v2 manual mostly as a placeholder for now. ]

In addition to users running a labyrinth within OpenLabyrinth itself, there is a web-service that allows activities to be run remotely. To run a remote OpenLabyrinth service you need to have a remote service (including a single client IP address) registered within OpenLabyrinth, one or more OpenLabyrinth maps associated with that service, and a means of presenting the activity to the user in the remote environment. This OpenLabyrinth ‘client’ application would need to be able to consume the XML generated by the service and render it so that the user was given the links and services required to run the labyrinth remotely and interact with the OpenLabyrinth server.

## Description

In addition to playing OpenLabyrinth activities within OpenLabyrinth itself, you can also run them in a remote system such as a virtual learning environment or e-assessment system by using OpenLabyrinth Remote Services. These services use XML to send OpenLabyrinth node content to the remote client that then communicates back using simple URLs (see figure 32).

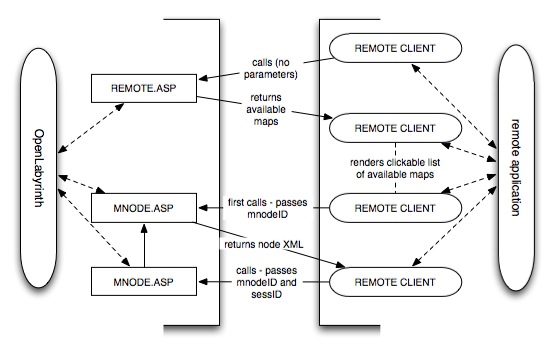


Figure 32: OpenLabyrinth Remote Services Architecture.

OpenLabyrinth is on the left and the remote application is on the right hand side. The steps are:

1. The remote client calls REMOTE.ASP in OpenLabyrinth
2. If the user credentials are correct or the remote client’s IP is recognised then REMOTE.ASP sends back an XML file listing the available Labyrinth Maps
3. The remote client parses the incoming XML into clickable links to start each Labyrinth Map
4. When the user clicks on a link this sends a call to MNODE.ASP passing just the nodeID
5. If the call is authorised then MNODE.ASP returns the node XML package
6. The remote client parses the incoming XML so the user can work through the activity. Each click cycles through steps 4-6
7. As this is happening additional communication can run within each application

## Setting up OpenLabyrinth Remote Services

There are two ways a remote service can be called:

1. by passing user credentials in the remote GET string. The user needs to be a remote type user (i.e. not a superuser, author or learner account). To set this up:

* create a remote user or reuse an existing one
* add the remote user to any maps you want to be available in the remote service
* call the remote service by adding hashed userID and password credentials (uid=xxx&pwd=yyy) using the cookieHash function in UTILITIES.ASP . There is a commented function in REMOTE.ASP that allows you to generate the hashed credentials

1. by registering a remote service. To set this up:

* click on ‘remote services’ on the home page and then ‘add a service’
* give the service a name and enter the IP address or range of the remote server. Leave box 4 or 3 and 4 blank to indicate a range of IP addresses, e.g. put 123 in box 1 and 45 in box 2 to accept any IP starting with 123.45.
* once the service has been created then add Labyrinth Maps to the service by clicking ‘add/edit Labyrinths’ and adding them from the dropdown menu

## Remote Services Components and Messaging

There are at least three discrete functions required at the client end:

1. Client list: this queries OpenLabyrinth for the OpenLabyrinth activities assigned to the current remote service and then formats and displays the results of the query
2. Client player: this interacts with the MNODE.ASP script in OpenLabyrinth to run the activity
3. Client remote services: these support additional OpenLabyrinth services like info boxes, counters and DAM nodes

In addition there may be local services layered on top of the remote OpenLabyrinth application. The components and their accompanying messaging flows are laid out in figure 33.



Figure 33: remote components and their interactions with OpenLabyrinth

## OpenLabyrinth Remote Services Transactions

There are three transaction types in the remote services: OpenLabyrinthservice, OpenLabyrinth and service.

### Transaction: OpenLabyrinthservice

This transaction is called by the client list function to get a list of the OpenLabyrinth activities available for the current remote service. The URL for the call is: http://OpenLabyrinth.mvm.ed.ac.uk/remote.asp

* <OpenLabyrinthservice> is the root element
* <remoteIP> is the IP address as seen by OpenLabyrinth
* <OpenLabyrinthmap> is the parent element per activity
* <OpenLabyrinthmapid> is the current activity’s map ID (integer)
* <OpenLabyrinthmapname> is the current activity’s title
* <OpenLabyrinthmaproot> is the root mnode ID (integer) for the current activity

As an example:

<?xml version="1.0"?>

<OpenLabyrinthservice>

<remoteIP>129.215.133.15</remoteIP><OpenLabyrinthmap>

<OpenLabyrinthmapid>174</OpenLabyrinthmapid>

<OpenLabyrinthmapname>ESSQ+1</OpenLabyrinthmapname>

<OpenLabyrinthmaproot>2054</OpenLabyrinthmaproot>

</OpenLabyrinthmap>

<OpenLabyrinthmap>

<OpenLabyrinthmapid>178</OpenLabyrinthmapid>

<OpenLabyrinthmapname>ESSQ+Jaundice+1</OpenLabyrinthmapname>

<OpenLabyrinthmaproot>2129</OpenLabyrinthmaproot>

</OpenLabyrinthmap>

</OpenLabyrinthservice>

### Transaction: OpenLabyrinth

This is the main transaction type with OpenLabyrinth which processes the activity as it plays out.

The URL for this is: http://OpenLabyrinth.mvm.ed.ac.uk/mnode.asp?id=x&mode=remote&sessID=y

The value for sessID can be left blank for the root node transaction but is required thereafter to maintain scores etc. As log as the node ID requested is in an activity that is allowed in the current service an XML response is sent back with the following elements:

* <OpenLabyrinth> is the root element
* <mnodetitle> is the current node’s URL-encoded title
* <mapname> is the current activity’s URL-encoded title
* <mapid> is the current activity’s (integer)
* <mnodeid> is the current node’s ID (integer)
* <mapscore> is the current activity score (integer)
* <message> is the current node’s URL-encoded body text
* <colourbar> is the current node’s URL-encoded text indicating the use of colour bars for zoned scores (in game mode only)
* <linker> is the current node’s URL-encoded HTML links
* <tracestring> is the current node’s URL-encoded HTML for the trace/track
* <rootnode> is the node ID for the current activity’s root node (integer)
* <infolink> is the current node’s ‘info’ button URL-encoded HTML
* <usermode> is a toggle value between basic and expert
* <dam> is a list of any associated MVP Data Availability Model nodes
* <mysession> is the GUID for this particular user session
* <counterstring> is URL-encoded text of links for all the counters in this map
* <timestring> and <javascripttime> are deprecated in the current version - disregard

As an example:

<?xml version="1.0"?>

<OpenLabyrinth>

<mnodetitle>Start</mnodetitle>

<javascripttime></javascripttime>

<mapname>ESSQ+Jaundice+1</mapname>

<mapid>178</mapid>

<mnodeid>2129</mnodeid>

<mapscore>100</mapscore>

<timestring></timestring>

<message>%3Cp%3Eyou+are+a+FY2+in+general+surgery+on+call++you+are+phoned+by+a+GP+who+has+a+55yo+woman++in+his+surgery+who+appears+jaundiced++what+do+you+do+now%3A%3C%2Fp%3E</message>

<colourbar>%3Ctable+border%3D%270%27+width%3D%27100%25%27+cellpadding%3D%275%27%3E%3Ctr%3E%3Ctd+height%</colourbar>

<linker>%3Cp%3E%3DABBCE08B%2D1F75%2D4A0A%2D852B%2D21F0819A19A2%27%3EID%3D%5B27%5D%5D%5D%5D+%2D+arrange+urgent+review++the+patient+comes+to+hospit%3C%2Fa%3E%3C%2Fp%3E</linker>

<tracestring>%3Ca+href%3D%22%23%22+onclick%3D%22toggle%5Fvisibility%28%27track%27%29%3B%22%3E%3Cp+class%3D%27style2%27%3E%3Cstrong%3EReview+your+pathway%3C%</tracestring>

<rootnode>2129</rootnode>

<infolink></infolink>

<usermode></usermode>

<dam></dam>

<mysession>ABBCE08B-1F75-4A0A-852B-21F0819A19A2</mysession>

<counterstring></counterstring>

</OpenLabyrinth>

### Transaction: service

This is a combined function that connects to three different OpenLabyrinth services:

* Information: http://OpenLabyrinth.mvm.ed.ac.uk/info.asp?id=x&mode=remote&sessid=y
* Counters: http://OpenLabyrinth.mvm.ed.ac.uk/counter.asp?id=x&cid=y&mode=remote&sessid=z
* DAM Nodes: http://OpenLabyrinth.mvm.ed.ac.uk/counter.asp?id=x&did=y&mode=remote&sessid=z

Each of these scripts bundles their response as a single XML element:

* <OpenLabyrinth> the root element
* <service> a URL-encoded HTML text response

The following info button call is an example:

<?xml version="1.0" encoding="UTF-8"?>

<OpenLabyrinth>

<service>Your+role+in+this+clinic+is+to+review+patients+on+behalf+of+your+Senior%2E+He+is+present+in+the+clinic%2C+and+will+call+in++to+see+if+you+are+progressing+well+in+the+case%2E+++This+is+the+Virtual+Hospital+Environment+%28VHE%29%2C+take+advantage+of+the+fact+there+is+no+time+pressure+and+make+sure+you+understand+the+basic+sciences+you+are+applying%2E+++Better+than+getting+a+textbook+out+in+front+of+the+patient%2E</service>

</OpenLabyrinth>

## Basic OpenLabyrinth Client Functions

The OpenLabyrinth client is relatively thin in that it only needs to send out appropriate and well-formed requests and to parse, format and present the returned XML data. As any free-text information is exchanged as hexadecimal data a converter from hex to plain text is required as is some data cleaning, for instance:

* Changing instances of "mnode.asp" to the local client URL.
* Changing instances of "/files/" to http://OpenLabyrinth.mvm.ed.ac.uk/files/ in any OpenLabyrinth resource URLs (such as images).
* Changing certain links so that they open in a new window.
* Changing instances of any HTML escaped characters back to their base format, for instance instances of "&apos;" should be turned back to an apostrophe.

## OpenLabyrinth Client Enhancements

The previous few sections have outlined the underlying basics of creating a remote OpenLabyrinth client service. It is possible however to build additional services on top of this as meet the requirements of the system in which the client service is instantiated. These services may be global to all surfaced OpenLabyrinth cases, they may be attached to specific activities (identified by their unique map ID) or to specific nodes (identified by their unique node ID).

Examples from existing clients include:

* The ability for users to make their own notes either at a particular node or throughout an activity
* The ability for users to record particular kinds/formats of data, for instance when working with a virtual patient it is useful to be able to record several differential diagnoses, to rank them and then to progressively whittle them down to a definite diagnosis as the activity unfolds
* Cross-integration and mashups with other dynamic web applications

# Customization: Skins and Mashups

## Skins

The skins control how the labyrinth is presented to the user. The skin for a particular labyrinth can be changed in the global editor. Standard installed skins include basic, PDA and NOSM, the latter is presented as a development template for users to develop their own OpenLabyrinth skins.

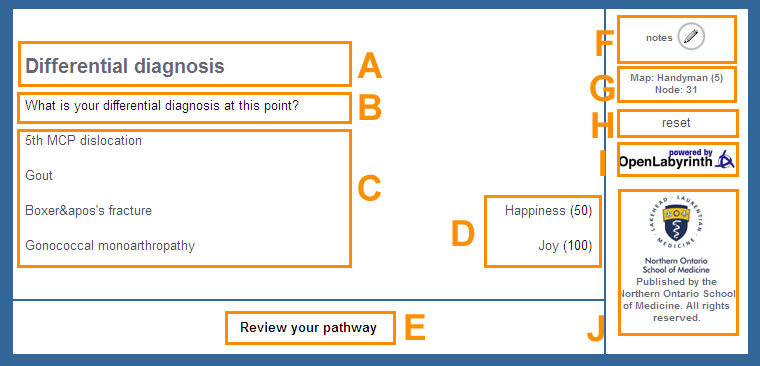


Figure 34: an example OpenLabyrinth skin (for Northern Ontario School of Medicine)

There are a number of different presentation elements in an OpenLabyrinth skin (see figure 34):

|  |  |  |
| --- | --- | --- |
| A: title  D: counters  G: metadata  J: skin-specific content | B: message  E: review session  H: reset session | C: options  F: notes  I: link to home page |

Effectively all that different skins do is to change the layout and visual presentation of these elements – see figure 35

|  |  |
| --- | --- |
| skin_stgeorges | pda skin |

Figure 35: some example OpenLabyrinth skins

To create a new skin:

* Create a new skin folder in the skins folder, you could start by duplicating an existing one
* Edit the ASP skin script and the accompanying CSS style sheet to change the HTML layout of the skin elements, adding new content as you require
* Add the skin reference to the SKINS table in the database – this will allow it to be presented as an option in the global editor

## Code

OpenLabyrinth has been written in an open and largely modular way to allow for new developments and changes to the code base. Moreover, OpenLabyrinth is provided as open source, which means that users are free to reuse and adapt the code to meet their own needs. You are strongly encouraged, although not required, to share any new code with the OpenLabyrinth community. Examples of code changes could include:

* New services such as tracking a differential diagnosis
* New data elements such as those for a different professional discipline
* Integration with other systems such as VLEs, assessment systems or

The homepage can also be customized, in particular there is a section below the



Figure 36: the OpenLabyrinth homepage

## Documentation

A copy of this user guide is also provided in PDF and Word DOC format. This can be amended and reused as set out in the Creative Commons Attribution-NonCommercial-Share Alike 3.0 license (see http://creativecommons.org/licenses/by-nc-sa/3.0/). For instance you could add new instructions, translate or adapt it for different contexts.

## Mashups

OpenLabyrinth remote web services can be used to create mashups with other remote applications. For instance:

* a labyrinth could be used to link with a wiki by pairing its nodes with the wiki pages
* a labyrinth could be used to link with other mashup applications such as Flickr, YouTube or various Google services

See section 8 for more information on using the OpenLabyrinth remote services. See section 20 for more information on connecting with other data systems.

# Development Techniques

This section pulls together some advice and tips on creating and using OpenLabyrinth activities.

## Using VUE

VUE is a free topic-mapping tool from Tufts University - see http://vue.tufts.edu/ for both Windows and Mac. You can use it to create designs for labyrinths by creating boxes to represent nodes and the links between them. Although VUE supports many other features only the boxes (converted to nodes) text in the boxes/nodes and the links (between nodes) will be imported but everything else will be ignored.

VUE allows a whole design to be viewed at once, and to be built in whatever way the author(s) wish (see Figure 37 for some examples). For instance a VUE design could be data projected for a group of authors to comment and build up a map collaboratively. Note that OpenLabyrinth v3 does not support direct import of VUE maps at present. You need to use an intermediate step of importing to an OpenLabyrinth2 server, exporting as an MVP package, which you then Import into OpenLabyrinth. If there is sufficient demand, we will create a VUE import tool for OpenLabyrinth as well.

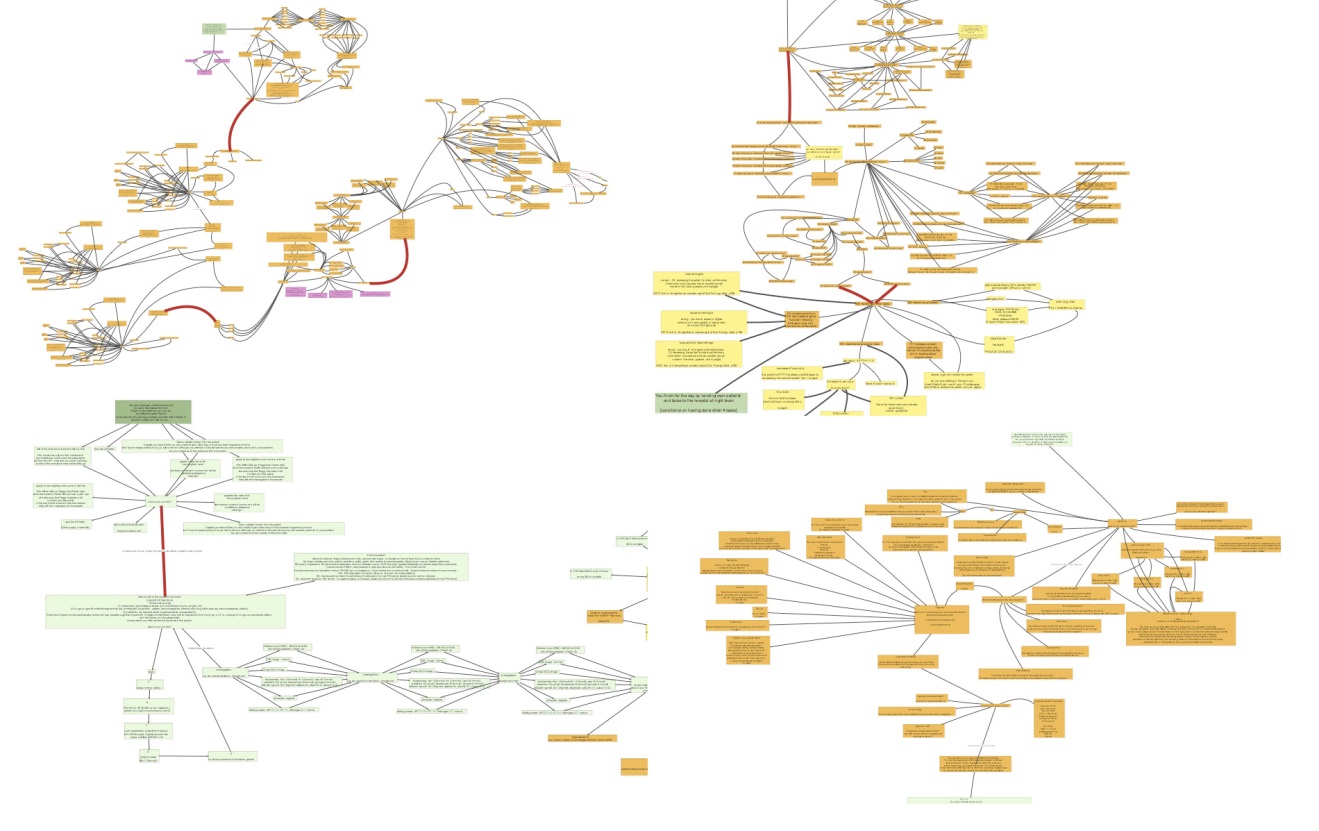
**

Figure 37: some labyrinth designs created in VUE

As well as creating labyrinth designs in a stepwise manner parts of other designs can be copy-pasted into new designs. For instance a design for an interview or making a diagnosis can be reused across a number of designs. Reusing these ‘design patterns’ can make development faster and more effective.

## Using Nodes

A node consists of a container for HTML content plus a number of rules and functions. A node may need to be duplicated if one or the other differs. Furthermore, a node may contain a media object that presents an activity in its own right, for instance an assessment question, an animation or an interactive object such as a puzzle or a mini game.

## Using Links

The links represent the topology of the labyrinth activity. You can use these links creatively to assemble different kinds of labyrinths:

* A randomly selected link can be used to represent a dice throw – make six links each for one of the dice values and then set the node to select just the one randomly. When the labyrinth is played the dice throws will be randomly selected each time, use alternative link text to hide what the value is.
* You can use alternative text and icons to control how options are presented to the user. For instance, you can allow users to select the image that best suits the question such as interpreting an x-ray or a histology image

You can use different link types to create more specific designs. For instance, Key Feature Problems (KFPs) can be designed using dropdown menus (Fischer et al 2005).

## Using Counters

You can use counters to track different kinds of values during an activity:

* Costs: a maximum financial budget can be set as the starting value and each choice that involves expenditure can have that cost deducted from the budget.
* Time: a time budget can be deducted from or arbitrary time added to based on the choices made. For instance, selecting a blood pressure test may take 3 minutes while a CT scan may take an hour.
* Psychological factors such as morale, general health or satisfaction can be tracked based on choices made.
* Clinical specific factors such as blood pressure, heart rate or respiratory rate can also be tracked against decisions made.
* Artefacts such as tools, drugs, equipment or keys can be acquired (value increased) and used (value decreased) using counters.

Remember that you can have any number of counters in any one labyrinth.

## Different Kinds of Labyrinth Designs

There are lots of different kinds of designs you can build in Labyrinth:

* Standard professional narratives such as history, examination, investigation, diagnosis and treatment.
* Narratives based on single cases or having to deal with multiple cases at once, each intruding or interrupting the other
* Algorithms such as clinical guidelines, diagnostic pathways for technical or user documentation.

# Frequently Asked Questions

In this user guide, we are only highlighting a few common issues. It is much better to maintain a FAQ list on a web site, where they can be more easily updated. Currently, these are hosted at <https://sites.google.com/site/openlabyrinth3/support_pages> but this is not a very effective system. In future, we will be moving to a more effective system. Other OpenLabyrinth author groups are encouraged to help this open-source project by contributing to this effort.

1. Why won't my case run?  
   The reason it would not play is that there was no starting point or root node set. For snippets, this would not be important but it does make it tough for you to see how it looks. I have set the top node as root and now you can see it Play.
2. Why does my case start in the wrong place?
3. What are these weird errors we see from Kohana?
4. Why do I have to pick from this limited set of choices?  
   There is no answer there that I like. I want to be able to type in my own answers.
5. Why won’t OpenLabyrinth run on my browser?
6. Where do I find good case examples?  
   See this section [Virtual Patient case libraries](#_Virtual_Patient_case_2)

# Further Information and Resources

OpenLabyrinth is provided as open source code and as such the authors accept that there may be some bugs and problems but make no guarantees and accept no liabilities for the same. If bugs are found then either fix them and share the code back to the OpenLabyrinth community or make a request for a fix to the community.

As for user support, there ain’t none – ‘they’ are ‘us’. OpenLabyrinth community members will generally help each other work through problems but only inasmuch as they have the ability and time to do so. As with any community it takes all members to work together to make the system stronger and better. Suggestions and ideas for new developments should also be circulated to the community.

OpenLabyrinth is licensed under the GNU General Public Licence v. 3.0 – see section 15 Appendix 2: GNU General Public License (GNU-GPL) v. 3.0.

## References

Begg, M., Dewhurst, D. and MacLeod, H. (2005). "Game Informed Learning: Applying computer game processes to Higher Education." Innovate 1(6).

Begg, M., Ellaway, R., Dewhurst, D. and Macleod, H. (2007). "Transforming Professional Healthcare Narratives into Structured Game-Informed-Learning Activities." Innovate 3(6).

Ellaway, R. (2004). "Modeling Virtual Patients and Virtual Cases." MELD. <http://meld.medbiq.org/primers/virtual_patients_cases_ellaway.htm>

Ellaway, R. (2007). Discipline Based Designs for Learning: The Example of Professional and Vocational Education. Design for Learning: rethinking pedagogy for the digital age. Beetham, H. and Sharpe, R., Routledge: pp153-165.

Fischer M., Kopp V, Holzer M,  Ruderich F, Jünger J (2005) “A modified electronic key feature examination for undergraduate medical students: validation threats and opportunities” [Medical Teacher](http://www.informaworld.com/smpp/title~content=t713438241~db=all), Volume 27, Issue 5, 2005, Pages 450 – 4

## Virtual Patient case libraries

We all benefit from sharing case materials openly and widely. There are a number of open repositories and libraries available. This is by no means a complete list. Indeed, because libraries are growing and expanding all the time, it is impossible to keep such a list up to date. Various efforts are under way to make cases more discoverable, including semantic linking using the mEducator2 SPARQL discovery engine. In the meantime, we encourage all virtual patient case authors to follow these steps to make your cases more available and more widely used:

* Make your case content open and modifiable using a Creative Commons Attribution-NonCommercial-ShareAlike license. <http://creativecommons.org/licenses/by-nc-sa/3.0/>
* Use an authoring tool that is compliant with the ANSI/Medbiq Virtual Patient standard, which makes it playable on a wider set of players.
* Make your repository searchable by semantic discovery tools such as SPARQL
* Post your cases not just to your own web site, but also to well known repositories such as MedEdPortal.

Case libraries that specifically contain OpenLabyrinth MVP format cases:

* <http://vp.openlabyrinth.ca/> - our own main server
* <http://www.virtualpatients.eu/> - eViP
* <http://pine.nosm.ca/> - Pathways in Narrative Education at NOSM
* <http://fmsharc.cfpc.ca/openlabyrinth/> - the SharcFM shared curriculum in Family Medicine, CFPC
* <http://demo.openlabyrinth.ca/> - our own development server, running the latest OpenLabyrinth code

If you know of other open libraries of compatible cases that should be included here, [please contact us](mailto:info@openlabyrinth.ca?subject=Suggestion%20for%20open%20VP%20case%20library%20(link%20from%20userguide)).

# Appendix 1: Installation

OpenLabyrinth is a web application written using PHP and Javascript. You need access to a server that is running a standard LAMP configuration (Linux, Apache,MySQL, PHP), the most common web platform out there, and all completely free and open source. OpenLabyrinth also requires a database joining the code. The most suitable database is MySQL Server. Note that the setup requires a certain amount of technical experience in modifying the server properties of a Linux platform. The whole service can be set up on virtual machine. Mac users can install it on the MAMP virtual machine and run a server on their own laptops.

The latest code and installation instructions can be freely accessed at <https://github.com/olab/Open-Labyrinth>

## Preparation

Provide a Linux server on which you will first need to install (if not already installed):

You may also need to install:

## Code and Directory

* Download the OpenLabyrinth package from https://github.com/olab/Open-Labyrinth and unzip its contents.

You have now put the code part of OpenLabyrinth in place. The next step is to set up the database.

## Database Setup

You installed MySQL or ensured you had a MySQL database available as part of the preparation step. Note that there are many factors that may change the nature of the following tasks – you should consult a database administrator for help if you get stuck.

Once the database is available:

* Add a new database called 'openlabyrinth'.

Once you have completed these steps that should be OpenLabyrinth all ready to go!

# Appendix 2: GNU General Public License (GNU-GPL) v. 3.0

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

## Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

## TERMS AND CONDITIONS

**0. Definitions.**

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

**1. Source Code.**

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

**2. Basic Permissions.**

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

**3. Protecting Users' Legal Rights From Anti-Circumvention Law.**

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

**4. Conveying Verbatim Copies.**

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

**5. Conveying Modified Source Versions.**

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
* b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
* c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
* d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

**6. Conveying Non-Source Forms.**

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
* b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
* c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
* d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
* e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

**7. Additional Terms.**

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
* b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
* c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
* d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
* e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
* f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

**8. Termination.**

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

**9. Acceptance Not Required for Having Copies.**

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

**10. Automatic Licensing of Downstream Recipients.**

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

**11. Patents.**

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

**12. No Surrender of Others' Freedom.**

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

**13. Use with the GNU Affero General Public License.**

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

**14. Revised Versions of this License.**

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

**15. Disclaimer of Warranty.**

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

**16. Limitation of Liability.**

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**17. Interpretation of Sections 15 and 16.**

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

# Collaborative editing as a team

It is not hard to put together a simple case as a solo author. However, for large projects, it is much better to form a team with various strengths to collaborate on various aspects of case writing. As well as dividing the work, it will also make your cases stronger as you can provide some internal peer review and suggestions about a case.

OpenLabyrinth was not originally conceived as a collaborative authoring environment. Indeed it is possible to seriously mess up each others’ work under certain circumstances so some caution is advisable.

## Potential team roles

* Clinician author  
  We find that having clinicians drive the main focus of a case works well in many situations. But remember that OpenLabyrinth cases do not have to be about patients.
* Learning designer or instructional designer  
  Not strictly necessary but it is good to have an expert keep your clinicians focused on good learning strategies.
* Copyright clearance and media discovery  
  This is essential if you are going to make your cases openly available. “Fair Use” provisions do not give educators freedom to use anything they find.
* Learners  
  It is too easy to lose sight of learner needs. Lots of value in including learners in your team.
* Graphic artist  
  While there are many wonderful images freely available out there, sometimes the concept you want to convey requires its own picture. Our most productive projects have benefited from an artist.
* Copy editor  
  Not usually required unless you are posting in a repository
* Peer reviewer  
  A very good idea. Makes your cases stronger.
* Programmer  
  Not really needed. But sometimes it helps to have someone who is more familiar with HTML coding.
* System administrator  
  Your web server will need to be hosted somewhere. OpenLabyrinth can be hosted on a large variety of web servers, including cloud and virtual servers. If your institution or group does not have access to its own server, try contacting one of the OpenLabyrinth servers in your region. Some will be happy to let you try some cases on their server. OpenLabyrinth does not need much administration once it has been installed, your SysAdmin will be pleased to hear.

## General cautions about collaborative editing in OpenLabyrinth

All case information in OpenLabyrinth is stored on a SQL database. But the OpenLabyrinth code has not been written with multi-user authoring in mind. There is no record locking in the database. User access control lists are pretty simple and do not prevent multiple authors who all have access to a case from overwriting each others’ work inadvertently. We are working on new components in OpenLabyrinth that will make it easier for teams to collaborate. In the meantime, bear in mind the following:

* Visual Editor, Node Grid, Counter Grid – these tools open up the whole case at once. If any one of the team is using one of these tools, it is very easy to overwrite work being done by other team members at the same time.
* If you notice that some nodes are changing while you are editing a case, be aware that another author is probably in the case. Avoid the 3 tools above. Try to instant message those whom you think might be in the case.
* Other parts of the OpenLabyrinth editing environment can be more safely used by team members at the same time. For example, if one team member is checking image sources for copyright, and another is creating Questions in the Questions editor, it is unlikely that their work will intersect. But again, be aware that OpenLabyrinth does not prevent two authors from overwriting each others’ work.
* Changes made by most parts of the OpenLabyrinth editing environment are usually written to the SQL database right away. This makes it easier to see your changes, even if you are in the middle of previewing or playing a case. But this same effect can throw off another player. Be careful.

# Syntax for Counter Rules

In OpenLabyrinth v2, authors had limited ability to regulate or direct the flow of the case narrative or logic. In OpenLabyrinth v3, this has been beefed up considerably by the provision of some simple conditional logic and action words. This is a work in progress and, at present, is quirky and frustrating to work with. But it does open up considerable power if you can be bothered to slog through these quirks.

This section has been left in the Appendices because we anticipate that it will grow considerably, as we provide better examples of how best to use this powerful feature.

Part of the problem of working with these Counter Rules is that currently OpenLabyrinth has an odd double pass through the counter rule. There are various reasons for this. We hope to streamline this aspect of OpenLabyrinth programming in the near future. This second pass through the Rules will often have unanticipated side effects.

## Event triggers and sequence of Counter evaluations

There are a number of events that can alter a Counter value when you are playing a case. It is helpful to understand the sequence in which these events are evaluated by the OpenLabyrinth code, if you are having trouble with getting your Counter Rules to work.

User clicks on linked option 🡪 Question in current node alters Counter value 🡪 User arrives at node 🡪 Node has static value assigned by Counter Grid 🡪 Counter Rule is evaluated as true 🡪 Counter Rule affects a Counter value 🡪 Counter Rule redirects user to another node

## Basic Syntax

IF.. THEN .. ELSEIF.. ELSE.. ENDIF

GOTO [[NODE:xxx]]

[[CR:nn]] =

Comparator: =, !=, <, <=, >, >=

MATCH([[CR:nn]], “Test string”)

UPPER(), LOWER(), PROPER(),

NO-ENTRY

Parentheses work but not consistently.

The Counter Rules editor has two modes of operation. Usually, it is easiest to use the ‘Text of rule’ tab where you simply use node titles and counter names. But if you are having trouble seeing why a rule is not being parsed correctly, the ‘Code of rule’ tab can be very useful.

We will provide more examples of Counter Rules and how to use them shortly.

# ImageMaps and hotspots

One interesting new feature in OpenLabyrinth v3 is the ability to directly enable hotspots in an image. This is not a new web technique at all, but previously it was a bit fiddly to do in OpenLabyrinth. The Node Editor and specifically, the TinyMCE editor, now directly allows you to create an area within an image that has an embedded hyperlink.

First you must use Node Editor to do this.

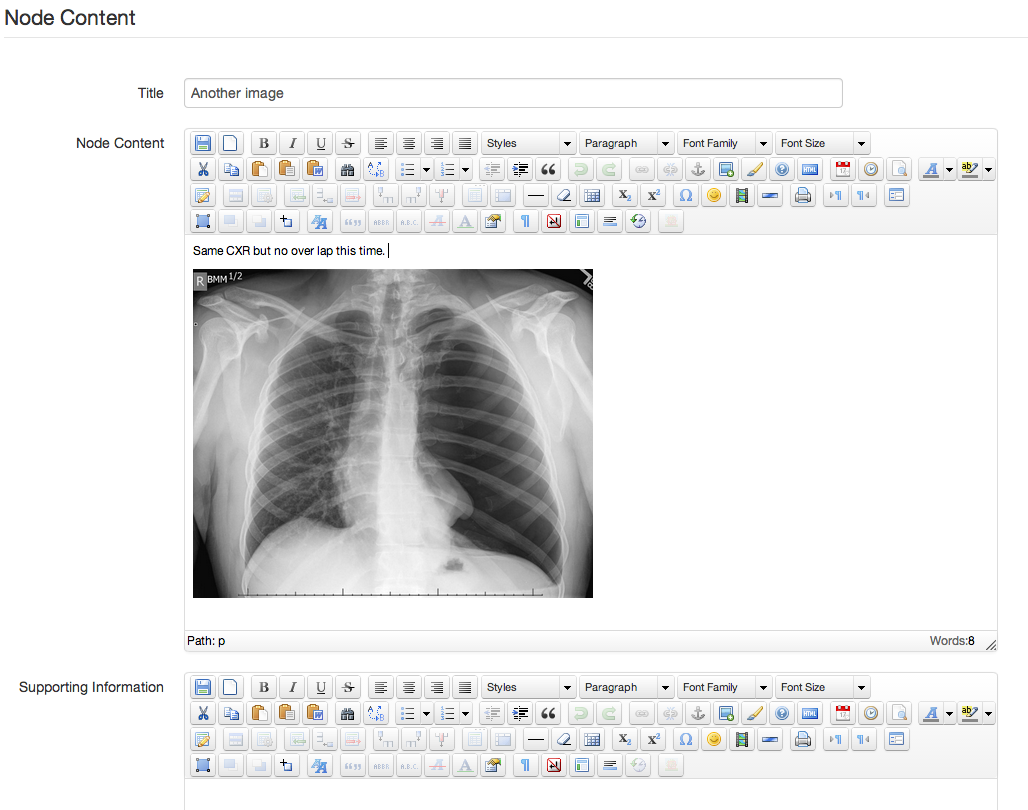


Figure 38: node editor page with a linked image

Normally when you are working with an image in OpenLabyrinth, you would insert the wiki style reference, like [[MR:123]. But in this case, we need to tell the TinyMCE editor that is part of the node editor more about the picture you are inserting.

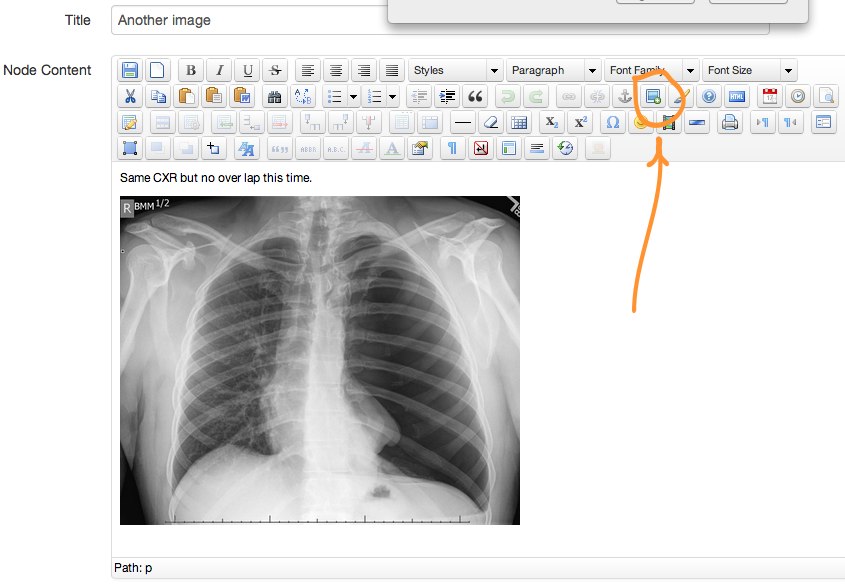


Figure 39: node editor page showing button to insert an image

First insert your image, using the image button. Provide a URL to the image. This is easy for images elsewhere on the web. For images in the Files section for the case, you can still do this but you need to insert the local link which will be something like ‘files/imageNameHere.jpg’ – if you have done it correctly, you will be able to see the picture itself in the TinyMCE editor. Next click on the ‘Image Map Editor’ button – see below

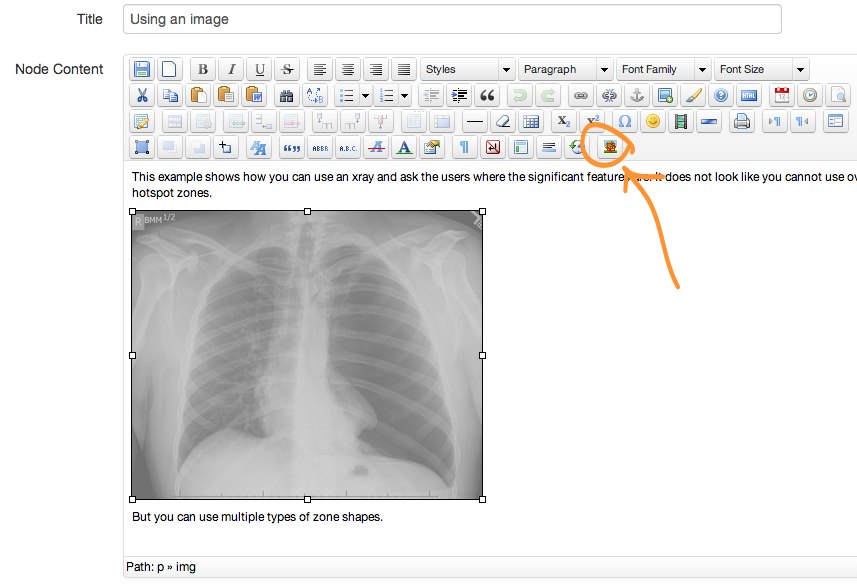


Figure 40: select the image in Node Editor then click ImageMap button shown

This will bring up the editing panel, with your picture in it. You can draw areas of interest, and attach a URL to that shape.

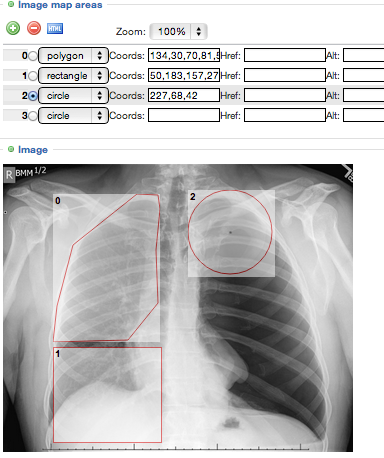


Figure 41: areas can be circles, rectangles or polygons

Simply outline the areas that you want to be active hyperlinks and then select the URL that you wish OpenLabyrinth to jump to. This can be another node in the case or outside of OpenLabyrinth entirely.

Remember to save when you are done.

With this technique, you can make key areas on your x-ray into teaching points – “spot the fracture” etc.

# Authentication systems

OpenLabyrinth has a simple, crude authentication and access control mechanism. For small groups, this is quite sufficient. But for those who are using these cases for multiple classes or who are dealing with many users, it is too cumbersome.

OpenLabyrinth has been designed so that it can be integrated with external authentication systems. For our initial work, we have used OpenLDAP as the example. LDAP is an old system but is very widely used. OpenLDAP is free and open-source. Reconfiguring OpenLabyrinth to work with another LDAP system should be easy.

There are other excellent authentication systems out there that we are also considering:

* OAuth
* SAML
* OpenID
* Shibboleth
* Active Directory

Each have their pro’s and con’s. Because OpenLabyrinth is a fully open-source software project, we invite other groups to make their implementations of connecting to other authentication systems more widely known so that we can all benefit from the various approaches to connecting OpenLabyrinth to other systems.

# Connecting with other systems

Increasingly, our data systems are being built so that they can share information with each other. This can be achieved in a number of ways. For example, OpenLabyrinth makes it data and services available via Web Services. See section [OpenLabyrinth Remote Services](#_OpenLabyrinth_Remote_).

We are looking at other options. We are frequently asked about SCORM compliance of our data objects. While it would be easy to apply one of two small items of metadata to our cases and call them SCORM compliant, this would not really make them more useful to you. SCORM was not designed for high levels of object interactivity and has largely become outmoded.

A more recent standard that affords data interchange between systems is IMS-LTI. We are currently working on making OpenLabyrinth LTI compliant.

Moving well beyond SCORM objects and basic static metadata, we are also building in the capacity for making our cases’ data discoverable by semantic indexing engines such as SPARQL via RDF.

In our HSVO project, we were able to connect directly to the HSVO Savoir platform, affording direct connectivity and discoverability with a number of other simulation devices. See <http://hsvo.org> for more information on this. We were also able to link OpenLabyrinth with the Laerdal™ SimMan3G API, passing data directly to and from their mannequins.

This is a rapidly evolving area of research. If you interested in collaborating on such data connectivity projects, [please contact us](mailto:info@openlabyrinth.ca?subject=Interested%20in%20collaborating%20on%20data%20connectivity%20projects%20and%20OLab3).

# Scenarios and Activity Designs

We have found OpenLabyrinth’s openness and flexibility is very useful for creating a variety of different activity designs, going way beyond isolated virtual patient cases.

* Group work cases
* Hybrid simulations with Standardized Patients
* Bookending high fidelity simulations
* Video Mashups
* Resource trade-off scenarios
* Multi-case triage
* Mini-cases as mobile exemplars
* Script Concordance Testing
* Bidirectional integration with podcasts and webinars

# Table of Figures

Figure 1*:* a typical labyrinth screen 7

Figure 2: a typical labyrinth screen’s elements 8

Figure 3: nodes and links 11

Figure 4: a node can show more than one InfoButton 13

Figure 5: edit InfoButtons using the Supporting Information field 14

Figure 6: first step in wizard to Create a case 15

Figure 7: select the kind of pattern you want to use 16

Figure 8: Select number of nodes then fill in bare essentials 17

Figure 9: using the Visual Editor to continue mapping the pattern 18

Figure 10: adding a small side branch using Visual Editor 19

Figure 11: add remaining components here 20

Figure 12: inline node editing: 21

Figure 13: the OpenLabyrinth properties editor, with common actions in sidebar on Left side. 22

Figure 14: Nodes panel with 'Add a node' button highlighted 24

Figure 15: the WYSIWYG tool bar 25

Figure 16: the Visual Editor 27

Figure 17: a typical labyrinth screen’s elements: 30

Figure 18: the Questions editor 31

Figure 19: starting off in creating a Question 32

Figure 20: inserting a Question into another labyrinth 33

Figure 21: the avatar editing screen 34

Figure 22: map files editor. 35

Figure 23: Counter Grid with six Counters 37

Figure 24: The relationship between VPS, MR, DAM and activity elements in OpenLabyrinth. 38

Figure 25 an embedded data element in the node editor (left) and how it renders on screen (right) 39

Figure 26: the matching editor and the way a matching question is presented to you 41

Figure 27: a typical OpenLabyrinth report histogram. 42

Figure 28: a typical counter trace 43

Figure 29: the add user screen 44

Figure 30: the interface.xml file that can be found in documents/interface.xml 45

Figure 31: the Presentations authoring screens 46

Figure 32: OpenLabyrinth Remote Services Architecture. 47

Figure 33: remote components and their interactions with OpenLabyrinth 49

Figure 34: an example OpenLabyrinth skin (for Northern Ontario School of Medicine) 52

Figure 35: some example OpenLabyrinth skins 53

Figure 36: the OpenLabyrinth homepage 54

Figure 37: some labyrinth designs created in VUE 55

Figure 38: node editor page with a linked image 71

Figure 39: node editor page showing button to insert an image 72

Figure 40: select the image in Node Editor then click ImageMap button shown 73

Figure 41: areas can be circles, rectangles or polygons 73

1. WYSIWYG = What You See Is What You Get, an old term, where the text appears just as you would see it on the page when the labyrinth is played. [↑](#footnote-ref-1)
2. TinyMCE is a platform independent web based Javascript HTML WYSIWYG editor control released as Open Source under [LGPL](http://labyrinth.mvm.ed.ac.uk/tinymce/jscripts/tiny_mce/license.txt) by Moxiecode Systems AB. Copyright © 2003-2006, [Moxiecode Systems AB](http://www.moxiecode.com/), All rights reserved. For more information visit the [TinyMCE website](http://tinymce.moxiecode.com/) at http://tinymce.moxiecode.com/ [↑](#footnote-ref-2)